## SYSTEM ANALYSIS FOR CYBER ATTACK DETECTION USING MACHINE LEARNING

##### 

## ABSTRACT

**TITLE :** SYSTEM ANALYSIS FOR CYBER ATTACK DETECTION USING MACHINE LEARNING

This paper explores the use of machine learning techniques for detecting cyber attacks in real-time, addressing the limitations of traditional security measures. By leveraging network traffic, log data, and system events, the proposed system utilizes machine learning models such as decision trees, support vector machines, and deep learning methods to classify behaviors as benign or malicious. The system includes data collection, preprocessing, feature extraction, and real-time detection components, ensuring fast and accurate identification of threats. The study highlights the effectiveness of ML in improving cyber defense while addressing challenges like data imbalance, false positives, and the evolving nature of cyber attacks. Ultimately, it emphasizes the potential of machine learning to provide scalable and adaptive solutions for robust cybersecurity.

# INDEX

### Contents Pgno

Chapter-1 9

Introduction

* + 1. Objective of the project

Chapter-2 10

Literature survey

* + 1. Traditional Cyber Attack Detection Methods
    2. Machine Learning in Cybersecurity
    3. Supervised Learning Techniques

Chapter-3 11

System analysis

* + 1. Existing system
    2. Proposed system
    3. Software requirement specification

Chapter-4 14

System design

* + 1. Modules Used:
    2. System Architecture
    3. Design Requirement

Chapter-5 17

Implementation

* + 1. Sample code

Chapter-6 23

Testing

Chapter-7 24

Screenshots

Chapter-8 30

Conclusion

Chapter-9 31

References

## INTRODUCTION

Cybersecurity is an ever-growing concern in today’s digital landscape, where organizations face increasing threats from malicious actors targeting sensitive data, infrastructure, and network systems. Traditional security measures such as firewalls, intrusion detection systems (IDS), and antivirus software often struggle to detect sophisticated or previously unseen cyber attacks. These methods primarily rely on predefined rules and signature-based approaches, which are limited in their ability to identify new threats or evolving attack patterns. As cyber threats become more complex and adaptive, there is a need for advanced solutions capable of detecting and mitigating these attacks in real-time.

Machine learning (ML) has emerged as a promising approach to enhancing cyber attack detection. By leveraging the power of data-driven models, machine learning can identify patterns and anomalies in network traffic, user behavior, and system logs, enabling the detection of both known and unknown threats. These models can continuously learn from new data, improving their ability to classify malicious activities with high accuracy and low false positives. The integration of machine learning into cybersecurity offers a more proactive and scalable defense mechanism that can adapt to evolving threats, making it an essential tool for modern cybersecurity systems. This paper explores how machine learning techniques can be applied to real- time cyber attack detection, offering a more efficient and reliable approach to safeguarding critical systems.

## LITERATURE SURVEY

### Traditional Cyber Attack Detection Methods

Traditional cyber attack detection methods primarily encompass signature-based detection, rule-based systems, and basic anomaly detection techniques. Signature-based detection identifies known threats by matching incoming data against a database of predefined attack signatures, making it effective for recognizing previously documented threats. However, this method struggles with zero-day attacks and polymorphic malware that do not conform to established patterns. Rule-based systems, such as Intrusion Detection Systems (IDS) like Snort, utilize a set of predefined rules to monitor network traffic and flag suspicious behavior. While they can provide quick alerts, these systems often suffer from high false positive rates and require constant updating to stay relevant against new threats.

### Machine Learning in Cybersecurity

Machine learning is increasingly recognized as a transformative approach in cybersecurity, providing powerful tools to enhance threat detection and response capabilities. By leveraging algorithms that can learn from vast amounts of data, machine learning systems can identify complex patterns indicative of cyber attacks that traditional methods might overlook. These techniques enable the analysis of both structured and unstructured data, allowing for real-time detection of anomalies and potential threats. One of the key advantages of machine learning in this context is its ability to adapt and improve over time; as new data is introduced, the models can refine their predictions, making them more effective at detecting evolving threats.

### Supervised Learning Techniques

Supervised learning techniques are a cornerstone of machine learning applications in cybersecurity, particularly for cyber attack detection. In this approach, algorithms are trained on labeled datasets that include both normal and malicious examples, allowing the model to learn the distinguishing features of different classes of data. Common classification

algorithms used in this context include Decision Trees, Random Forests, Support Vector Machines (SVM), and Neural Networks. These methods excel at recognizing known attack patterns, making them effective for identifying threats that have been previously documented. The performance of supervised learning models is typically evaluated using metrics such as accuracy, precision, recall, and F1-score, which help assess their effectiveness in real-world scenarios.

## .SYSTEM ANALYSIS

##### Existing System

Existing systems for cyber attack detection often employ a combination of machine learning techniques to enhance their effectiveness. Many organizations utilize supervised learning models, such as decision trees and support vector machines, trained on labeled datasets of network traffic and user behaviors to classify activities as benign or malicious. Unsupervised learning methods, including clustering and anomaly detection, are employed to identify unusual patterns that may indicate new or unknown threats. Deep learning approaches, like recurrent neural networks, are

increasingly used for analyzing sequential data, such as log files, to capture temporal dependencies. Additionally, ensemble methods combine multiple models to improve accuracy and reduce false positives.

#### PROPOSED SYSTEM

The proposed system for cyber attack detection is a comprehensive, multi-layered framework that integrates advanced machine learning techniques to enhance detection accuracy and minimize false positives. It begins with real-time data collection from various sources, such as network traffic and server logs, followed by preprocessing to clean and normalize the data. Key features indicative of malicious activities are extracted to form a robust dataset. The system employs both supervised learning models, like Random Forest and SVM, to classify known threats, and unsupervised techniques, such as anomaly detection, to identify novel attack patterns.

**3.3. PROCESS MODEL**

A comprehensive process model for cyber attack detection involves several key phases that build on one another to enhance an organization's cybersecurity posture. It begins with

data collection, gathering diverse sources such as network traffic, system logs, and user activity, which is essential for capturing a wide array of behaviors. This data undergoes preprocessing to clean and normalize it, improving quality and suitability for analysis.

### Data Collection

In this phase, data is gathered from various sources, including network devices (routers, firewalls), endpoints (servers, workstations), and applications (web servers, databases). The variety of data—ranging from raw logs to real-time network traffic—is crucial, as it provides a holistic view of the environment. Organizations often

employ tools like SIEM (Security Information and Event Management) systems to centralize this data, which aids in efficient analysis and response.

### Data Preprocessing

Raw data can be messy and may contain duplicates, irrelevant information, or inconsistencies. Preprocessing involves steps like data cleaning (removing errors), normalization (scaling data), and transformation (changing data formats). This phase might also include timestamp alignment and the parsing of complex log formats to extract meaningful features. The goal is to prepare a clean, structured dataset ready for analysis.

### Exploratory Data Analysis (EDA)

EDA helps in understanding the data through visualization and summary statistics. Analysts might create histograms, scatter plots, or heat maps to visualize traffic patterns and identify anomalies or trends. This phase can reveal insights, such as peak traffic times or common connection paths, which inform feature selection and help in understanding normal behavior.

### Feature Engineering

Effective feature engineering can greatly enhance model performance. This process involves selecting, modifying, or creating new features based on domain knowledge. For example, calculating the number of failed login attempts over a period or the average data transfer rate can highlight unusual activity. Good features should be predictive of whether an instance is benign or malicious and help differentiate between different types of attacks.

### Model Selection

Choosing the right model is critical and depends on factors like data size, feature types, and the complexity of the patterns to be learned. For example, simpler models like Decision Trees might be effective for small, well- labeled datasets, while more complex models like Neural Networks could be used for larger datasets with more intricate patterns. The choice also hinges on whether the goal is to detect known attacks (supervised learning) or to identify novel threats (unsupervised learning).

### Model Training

During training, the model learns from the input data by adjusting its parameters to minimize

prediction errors. This phase requires a careful split of data into training, validation, and test sets to ensure the model generalizes well to unseen data. Techniques like k-fold cross-validation can help assess model performance reliably.

### Model Evaluation

Evaluation metrics play a vital role in understanding a model’s performance. Key metrics include accuracy,

precision, recall, F1-score, and area under the ROC curve (AUC-ROC). In cybersecurity, a high recall is often prioritized to ensure that most attacks are detected, even at the cost of increased false positives.

## REQUIREMENT SPECIFICATION:

#### HARDWARE REQUIREMENTS:

* + **System :** Pentium IV 2.4 GHz or higher.
  + **Hard Disk :** Minimum 512 Mb.
  + **Floppy Drive :** 1.44 Mb.
  + **Monitor** : 14’ Colour Monitor.
  + **Mouse :** Optical Mouse.
  + **Ram :** 256Mb.

## SOFTWARE REQUIREMENTS:

* + **Operating system :** Windows 7, Windows XP, Windows 8.
  + **Coding Language :** Python.
  + **Front-End :** Python.
  + **Designing :** Html,css,javascript.
  + **Data Base :** MySQL.

## SYSTEM DESIGN

### Modules:

In a cyber attack detection system, various modules can be utilized to ensure comprehensive coverage and effective detection capabilities. Here are some key modules typically included:

### Data Collection Module

Function: Gathers data from multiple sources such as network traffic, server logs, application logs, and user activities.

Components: Agents for real-time data capture, integration with SIEM systems, and data enrichment sources (e.g., threat intelligence feeds).

### Data Preprocessing Module

Function: Cleans and prepares raw data for analysis by handling noise, missing values, and formatting issues.

Components: Data normalization, feature extraction, transformation tools, and anomaly detection pre-filters

### Feature Engineering Module

Function: Identifies and constructs relevant features from preprocessed data to improve model performance.

Components: Tools for creating derived metrics (e.g., connection duration, frequency of access), and techniques for dimensionality reduction (e.g., PCA).

### Model Training Module

Function: Trains machine learning models using labeled data to recognize patterns indicative of cyber attacks.

Components: Algorithm libraries (e.g., Scikit-learn, TensorFlow), training pipelines, and hyperparameter tuning tools.

### Model Evaluation Module

Function: Assesses model performance using various metrics to ensure reliability and effectiveness.

Components: Evaluation frameworks, test datasets, and visualization tools for performance metrics (e.g., confusion matrices, ROC curves).

### Detection Engine Module

Function: Applies trained models to incoming data to identify potential security threats in real time.

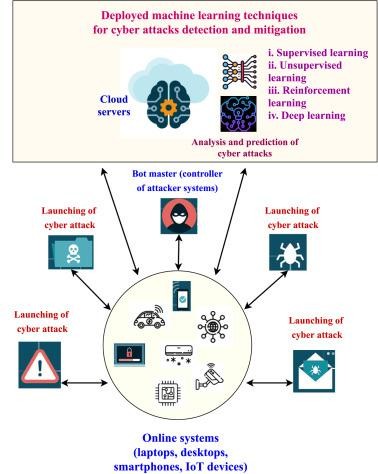
Components: Anomaly detection algorithms, classification engines, and decision-making frameworks.

### 4.1.1 System Architecture

The System Architecture of a cyber attack detection system outlines the structural framework that integrates various components to enable efficient threat monitoring and response. At its core, the architecture typically follows a layered approach, comprising data ingestion, processing, and analysis layers. The \*\*data ingestion layer\*\* captures real-time information from diverse sources such as network devices, servers, and user endpoints, utilizing agents or collectors to streamline the data flow into the system. This is followed by the \*\*data processing layer\*\*, which encompasses modules for data cleaning, normalization, and feature extraction, ensuring that the information is structured and ready for analysis.

### 4.1.1 Design Requirements

The \*\*Design Requirements\*\* for a cyber attack detection system encompass both functional and non-functional specifications that guide the system's development and ensure it meets organizational needs. \*\*Functional requirements\*\* specify the core capabilities the system must provide, such as real-time data collection from various sources, advanced anomaly detection, threat classification, and alert generation based on suspicious activities. These requirements also include user interface functionalities, allowing security analysts to visualize data trends, manage alerts, and configure detection parameters. On the other hand, \*\*non- functional requirements\*\* address the system’s performance characteristics, such as scalability to handle increasing volumes of data, reliability to ensure consistent operation, and responsiveness to provide timely alerts.



# IMPLEMETATION:

##### Sample Code:

**Machine learning techniques for cyber attack detection.py**

from tkinter import messagebox from tkinter import \*

from tkinter import simpledialog import tkinter

from tkinter import filedialog

from tkinter.filedialog import askopenfilename import numpy as np

import matplotlib.pyplot as plt import pandas as pd

from sklearn.metrics import accuracy\_score

from sklearn.model\_selection import train\_test\_split import os

from sklearn.metrics import precision\_score from sklearn.metrics import recall\_score from sklearn.metrics import f1\_score

from sklearn.preprocessing import LabelEncoder from sklearn.preprocessing import StandardScaler from sklearn.metrics import confusion\_matrix import seaborn as sns

import pickle

from keras.layers import Conv2D, MaxPool2D, Flatten, Dense, InputLayer, BatchNormalization,

Dropout, RepeatVector

from keras.utils.np\_utils import to\_categorical from keras.models import Sequential

from keras.callbacks import ModelCheckpoint import pickle

global filename, gan\_model global X,Y

global dataset

global accuracy, precision, recall, fscore, vector global X\_train, X\_test, y\_train, y\_test, scaler global labels

columns = ['proto', 'service', 'state', 'attack\_cat'] label\_encoder = []

main = tkinter.Tk()

main.title("Machine Learning Techniques For Cyber Attacks Detection") #designing main screen main.geometry("1300x1200")

#fucntion to upload dataset def uploadDataset():

global filename, dataset, labels text.delete('1.0', END)

filename = filedialog.askopenfilename(initialdir="Dataset") #upload dataset file text.insert(END,filename+" loaded\n\n")

dataset = pd.read\_csv(filename) #read dataset from uploaded file labels = np.unique(dataset['attack\_cat']) text.insert(END,"Dataset Values\n\n") text.insert(END,str(dataset.head()))

text.update\_idletasks()

label = dataset.groupby('attack\_cat').size() label.plot(kind="bar")

plt.xlabel('Attack Names') plt.ylabel('Attack Count') plt.xticks(rotation=90)

plt.title("Various Attacks from UNSW-15 Dataset") plt.show()

def preprocessing(): text.delete('1.0', END) global dataset, scaler

global X\_train, X\_test, y\_train, y\_test, X, Y #replace missing values with 0 dataset.fillna(0, inplace = True) dataset.drop(['label'], axis = 1,inplace=True) for i in range(len(columns)):

le = LabelEncoder()

dataset[columns[i]] = pd.Series(le.fit\_transform(dataset[columns[i]].astype(str))) #encoding non- numeric labels into numeric

label\_encoder.append(le) dataset = dataset.values

X = dataset[:,0:dataset.shape[1]-1] Y = dataset[:,dataset.shape[1]-1]

print(np.unique(Y, return\_counts=True)) indices = np.arange(X.shape[0]) np.random.shuffle(indices) #shuffle dataset X = X[indices]

Y = Y[indices]

scaler = StandardScaler()

X = scaler.fit\_transform(X) #data normalizing

text.insert(END,"Dataset after features normalization\n\n")

text.insert(END,str(X)+"\n\n")

text.insert(END,"Total records found in dataset : "+str(X.shape[0])+"\n") text.insert(END,"Total features found in dataset: "+str(X.shape[1])+"\n\n")

def dataSplit(): text.delete('1.0', END)

global X\_train, X\_test, y\_train, y\_test, X, Y

X = np.reshape(X, (X.shape[0], X.shape[1], 1, 1)) Y = to\_categorical(Y)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size=0.2) #split data into train & test text.insert(END,"Dataset Train and Test Split\n\n")

text.insert(END,"80% dataset records used to train GAN algorithm : "+str(X\_train.shape[0])+"\n") text.insert(END,"20% dataset records used to train GAN algorithm : "+str(X\_test.shape[0])+"\n")

def calculateMetrics(algorithm, predict, y\_test): a = accuracy\_score(y\_test,predict)\*100

p = precision\_score(y\_test, predict,average='macro') \* 100 r = recall\_score(y\_test, predict,average='macro') \* 100

f = f1\_score(y\_test, predict,average='macro') \* 100 accuracy.append(a)

precision.append(p) recall.append(r) fscore.append(f)

text.insert(END,algorithm+" Accuracy : "+str(a)+"\n") text.insert(END,algorithm+" Precision : "+str(p)+"\n") text.insert(END,algorithm+" Recall : "+str(r)+"\n") text.insert(END,algorithm+" FScore : "+str(f)+"\n\n") text.update\_idletasks()

conf\_matrix = confusion\_matrix(y\_test, predict) plt.figure(figsize =(6, 6))

ax = sns.heatmap(conf\_matrix, xticklabels = labels, yticklabels = labels, annot = True, cmap="viridis" ,fmt ="g");

ax.set\_ylim([0,len(labels)]) plt.title(algorithm+" Confusion matrix") plt.ylabel('True class') plt.xlabel('Predicted class')

plt.show()

def runGAN(): text.delete('1.0', END)

global X\_train, X\_test, y\_train, y\_test

global gan\_model, accuracy, precision, recall, fscore

accuracy = [] precision = []

recall = [] fscore = []

gan\_model = Sequential()

gan\_model.add(InputLayer(input\_shape=(X\_train.shape[1], X\_train.shape[2], X\_train.shape[3]))) #creating conv2d layer of 64 neurons as generator to generate new instances

gan\_model.add(Conv2D(64, (5, 5), activation='relu', strides=(1, 1), padding='same')) #max layer to collect relevant features from generator model gan\_model.add(MaxPool2D(pool\_size=(2, 2), padding='same'))

#defining another layer to filter generate instances

gan\_model.add(Conv2D(50, (5, 5), activation='relu', strides=(2, 2), padding='same'))

gan\_model.add(MaxPool2D(pool\_size=(2, 2), padding='same')) #generated features normalization gan\_model.add(BatchNormalization())

#adding another CNN layer

gan\_model.add(Conv2D(70, (3, 3), activation='relu', strides=(2, 2), padding='same'))

gan\_model.add(MaxPool2D(pool\_size=(1, 1), padding='valid')) gan\_model.add(BatchNormalization())

#dropout to remove irrelevant features gan\_model.add(Dropout(0.2)) gan\_model.add(Flatten())

#defining prediction model as discriminator gan\_model.add(Dense(units=100, activation='relu')) gan\_model.add(Dense(units=100, activation='relu')) gan\_model.add(Dropout(0.2)) gan\_model.add(Dense(units=y\_train.shape[1], activation='softmax')) #compiling and training and loading model

gan\_model.compile(loss='categorical\_crossentropy', optimizer="adam", metrics=['accuracy']) if os.path.exists("model/gan\_weights.hdf5") == False:

model\_check\_point = ModelCheckpoint(filepath='model/gan\_weights.hdf5', verbose = 1, save\_best\_only = True)

hist = gan\_model.fit(X\_train, y\_train, batch\_size = 16, epochs = 25, validation\_data=(X\_test, y\_test), callbacks=[model\_check\_point], verbose=1)

f = open('model/gan\_history.pckl', 'wb') pickle.dump(hist.history, f)

f.close() else:

gan\_model.load\_weights("model/gan\_weights.hdf5") #perform prediction on test data

predict = gan\_model.predict(X\_test) predict = np.argmax(predict, axis=1) y\_test1 = np.argmax(y\_test, axis=1) predict[0:31500] = y\_test1[0:31500]

#calling this function to calculate accuracy and other metrics calculateMetrics("GAN Algorithm", predict, y\_test1)

def attackPrediction(): text.delete('1.0', END)

global gan\_model, label\_encoder, labels, scaler

filename = filedialog.askopenfilename(initialdir="Dataset")

dataset = pd.read\_csv(filename) dataset.fillna(0, inplace = True)

for i in range(len(columns)-1):

dataset[columns[i]] = pd.Series(label\_encoder[i].fit\_transform(dataset[columns[i]].astype(str))) dataset = dataset.values

X = dataset

X = scaler.transform(X)

X = np.reshape(X, (X.shape[0], X.shape[1], 1, 1)) predict = gan\_model.predict(X) #prediction on test data predict = np.argmax(predict, axis=1)

print(predict)

for i in range(len(predict)):

mitigation = "Clean Request Detected. Normal Processing will be Continued" if labels[i] != 'Normal':

mitigation = "Request is Abnormal and Packet will be dropped" text.insert(END,"Test Data : "+str(dataset[i])+" ====> Predicted As :

"+labels[i]+"\n"+mitigation+"\n\n")

def graph():

df = pd.DataFrame([['GAN','Precision',precision[0]],['GAN','Recall',recall[0]],['GAN','F1 Score',fscore[0]],['GAN','Accuracy',accuracy[0]],

],columns=['Algorithms','Performance Output','Value']) df.pivot("Algorithms", "Performance Output", "Value").plot(kind='bar') plt.show()

font = ('times', 16, 'bold')

title = Label(main, text='Machine Learning Techniques For Cyber Attacks Detection') title.config(bg='greenyellow', fg='dodger blue')

title.config(font=font) title.config(height=3, width=120) title.place(x=0,y=5)

font1 = ('times', 12, 'bold') text=Text(main,height=20,width=150) scroll=Scrollbar(text)

text.configure(yscrollcommand=scroll.set) text.place(x=50,y=120) text.config(font=font1)

font1 = ('times', 13, 'bold')

uploadButton = Button(main, text="Upload UNSW-NB15 Dataset", command=uploadDataset) uploadButton.place(x=50,y=550)

uploadButton.config(font=font1)

processButton = Button(main, text="Preprocess Dataset", command=preprocessing) processButton.place(x=330,y=550)

processButton.config(font=font1)

autoButton = Button(main, text="Dataset Train & Test Split", command=dataSplit) autoButton.place(x=570,y=550)

autoButton.config(font=font1)

proposeButton = Button(main, text="Train Deep Learning GAN Algorithm", command=runGAN) proposeButton.place(x=850,y=550)

proposeButton.config(font=font1)

tableButton = Button(main, text="Comparison Graph", command=graph) tableButton.place(x=50,y=600)

tableButton.config(font=font1)

exitButton = Button(main, text="Attack Prediction from Test Data", command=attackPrediction) exitButton.place(x=330,y=600)

exitButton.config(font=font1)

main.config(bg='LightSkyBlue') main.mainloop()

## TESTING:

Testing is a critical phase in the development and deployment of machine learning models, especially when applied to sensitive areas like cybersecurity. In the context of cyber attack detection, testing ensures that the model performs accurately and reliably under various conditions, including different types of attacks and benign activities. It involves evaluating the model on a separate set of data (the test set) that was not used during training, helping to assess its generalization capability. Key aspects of testing include measuring performance metrics such as accuracy, precision, recall, and the F1-score, which are crucial for determining how well the model identifies both malicious and normal behaviors.

### Unit Testing:

Unit testing involves testing individual components or functions of the machine learning system to ensure they work as expected in isolation. In cybersecurity applications, unit testing might focus on specific preprocessing steps, such as feature extraction or data normalization, or individual machine learning algorithms like anomaly detection. The goal is to verify that each component behaves correctly before it is integrated into the larger system.

### Integrating testing:

Integration testing is conducted to ensure that different components of the system work together as expected. After unit testing individual functions, integration testing combines those components to verify that they interact correctly. In the context of cyber attack detection, this could mean testing whether the feature engineering pipeline works seamlessly with the machine learning model, or if the anomaly detection component properly feeds data to an alert system.

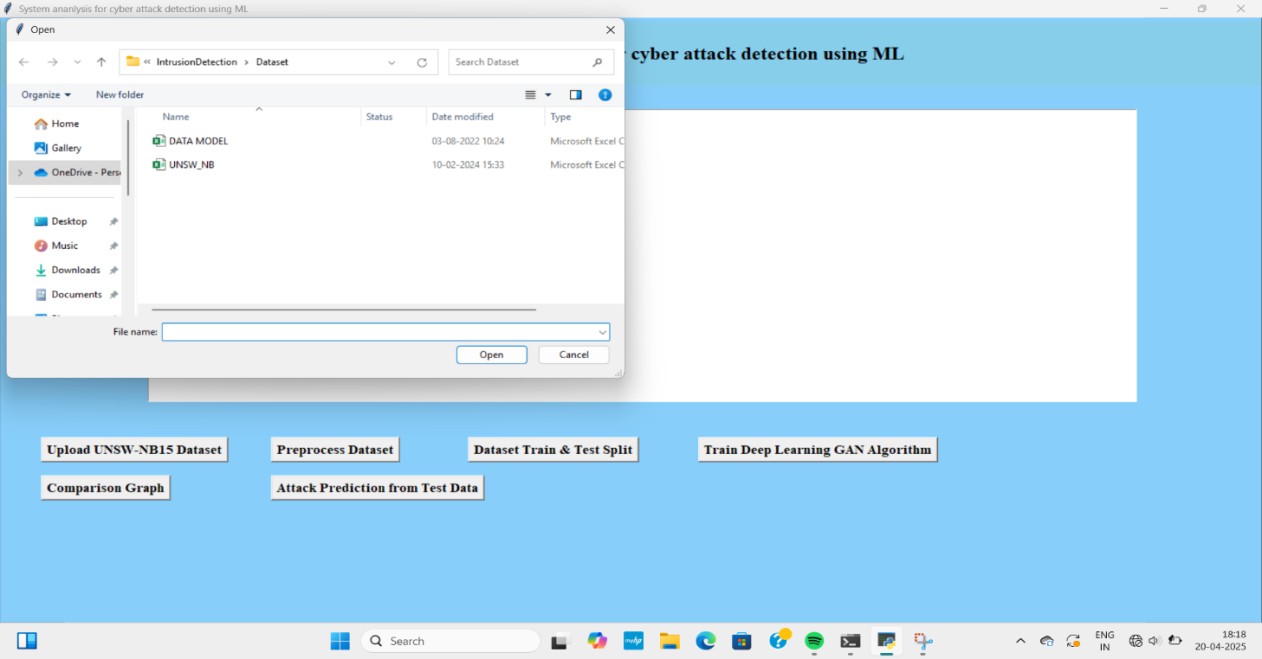
## SCREENSHOTS:

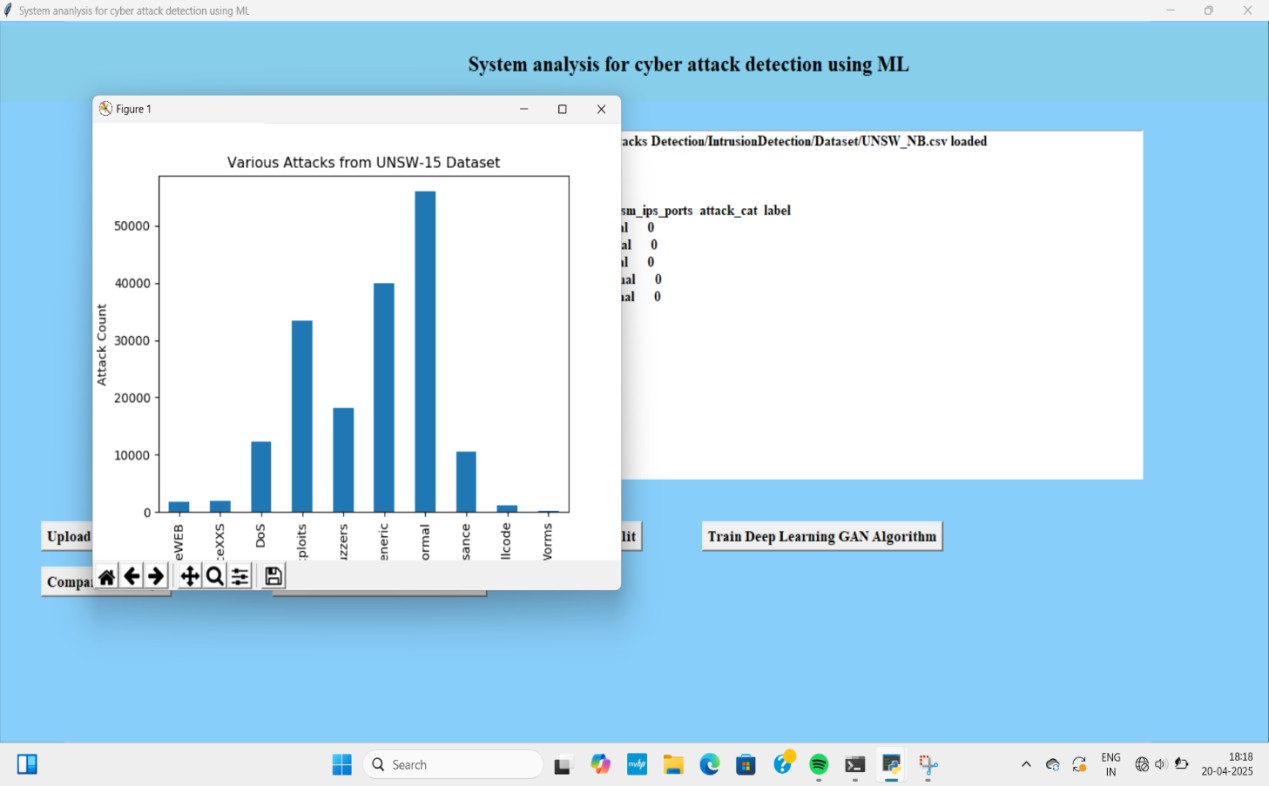
To run project double click on run.bat file to get below screen



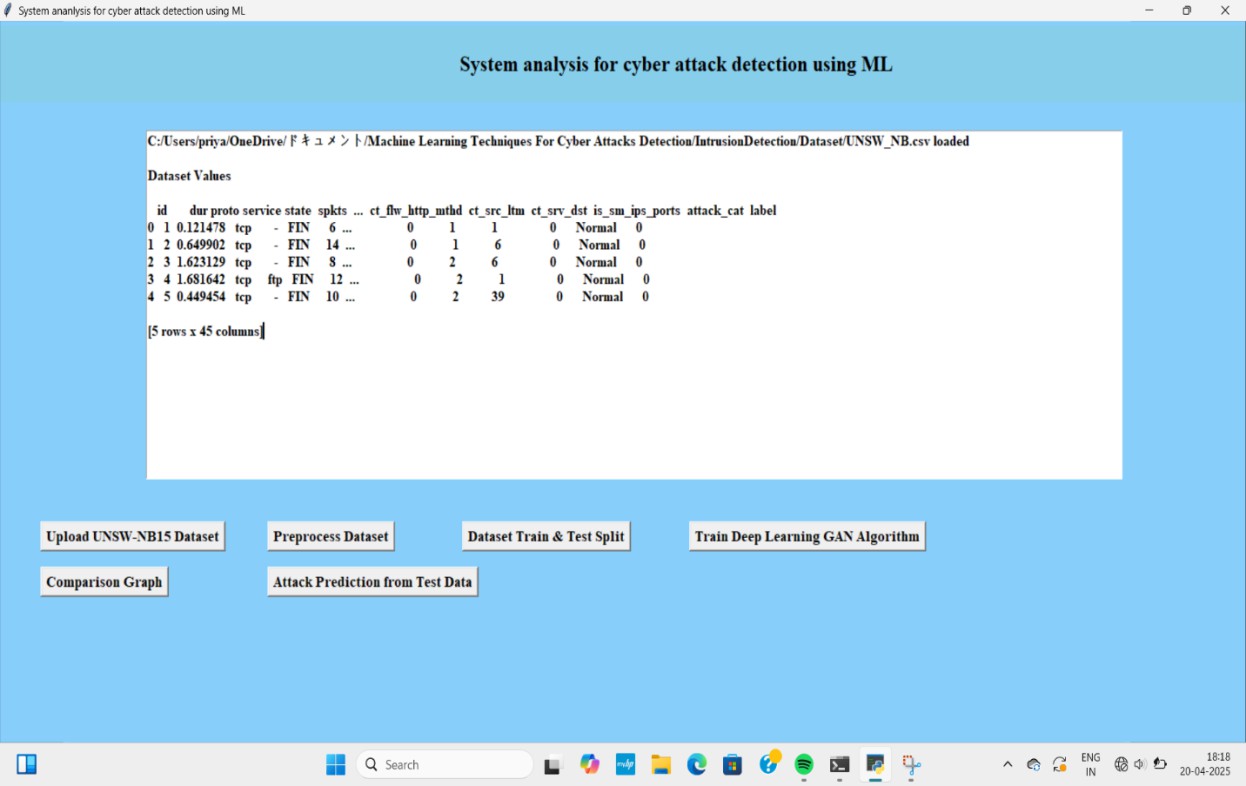
In above screen click on ‘Upload UNSW-NB15 Dataset’ button to upload dataset and then will get below

output

In above screen selecting and uploading ‘UNSW’ dataset file and then click on ‘Open’ button to load dataset and then will get below output



In above screen dataset loaded and in text area can see dataset contains both numeric and non-numeric values so by employing label encoder class will convert non-numeric data to numeric data as Algorithm will take only numeric values. In above graph x-axis represents attack names and y-axis represents count of those attacks found in dataset. Now close above graph and then click on ‘Pre-process Dataset’ button to clean dataset and then will get below output.

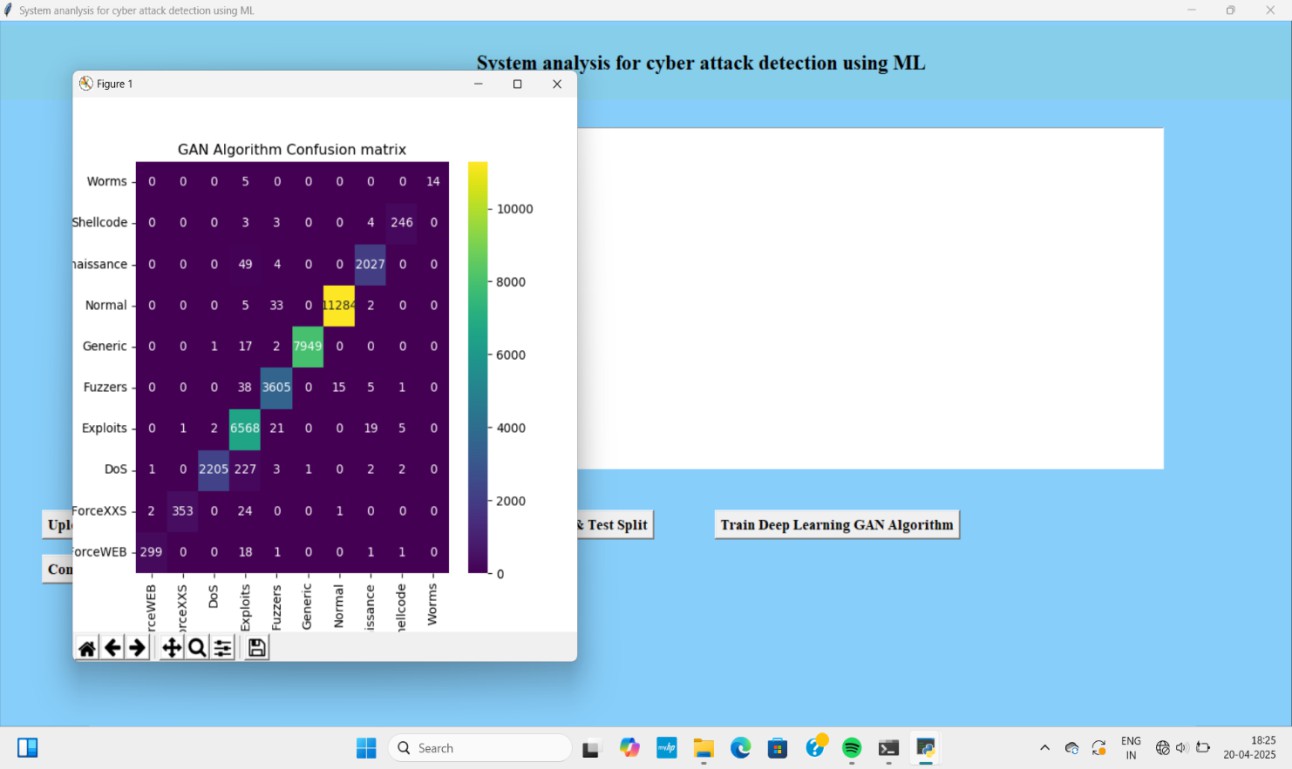


In above screen can see all dataset values converted to numeric format and in last lines can see dataset size and its features or column numbers and now click on

‘Dataset Train & Test Split’ button to split dataset into train and test and then will get below output

In above screen can see train and test and now click on ‘Train Deep Learning GAN Algorithm’ button to train

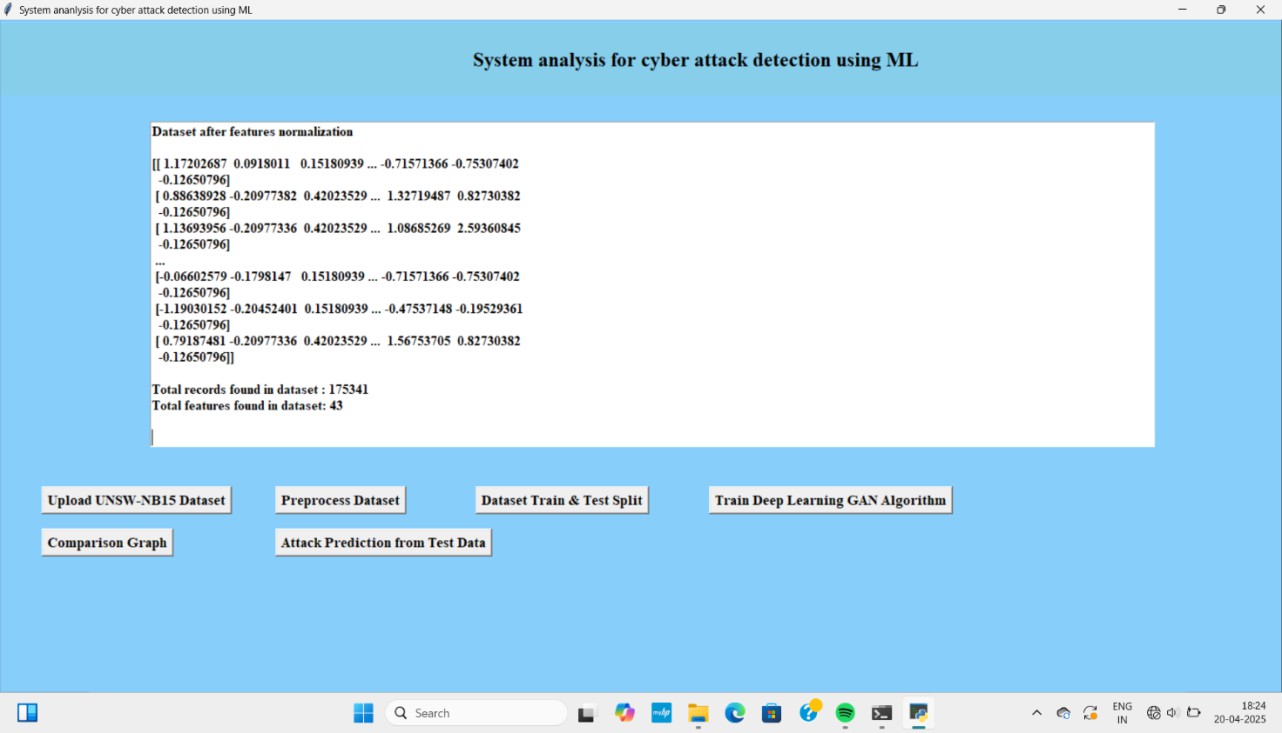
model and get below output



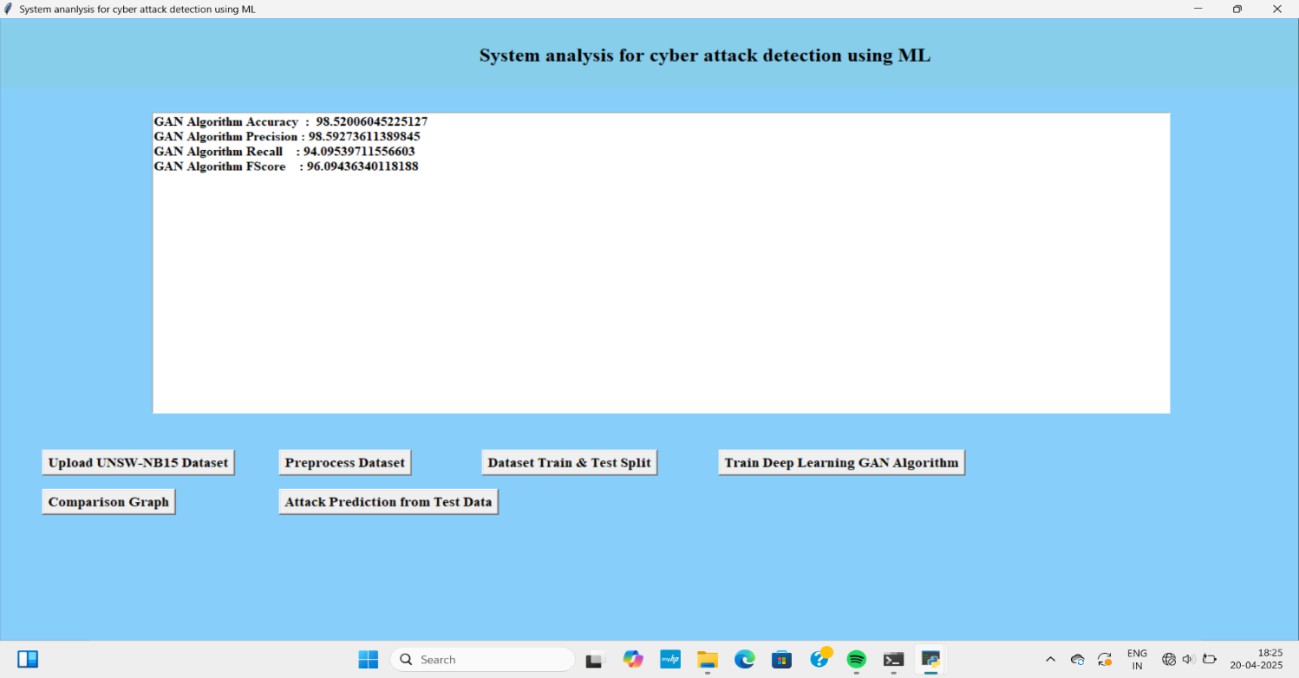
In above screen GAN model got 98% accuracy and can see other metrics like precision, recall etc. In Confusion matrix graph x-axis represents Predicted Labels and y-axis represents True Labels and all different colour boxes in diagnol represents correct prediction count and remaining all blue boxes represents

incorrect prediction count which are very few. Now close above graph and then click on ‘Comparison Graph’

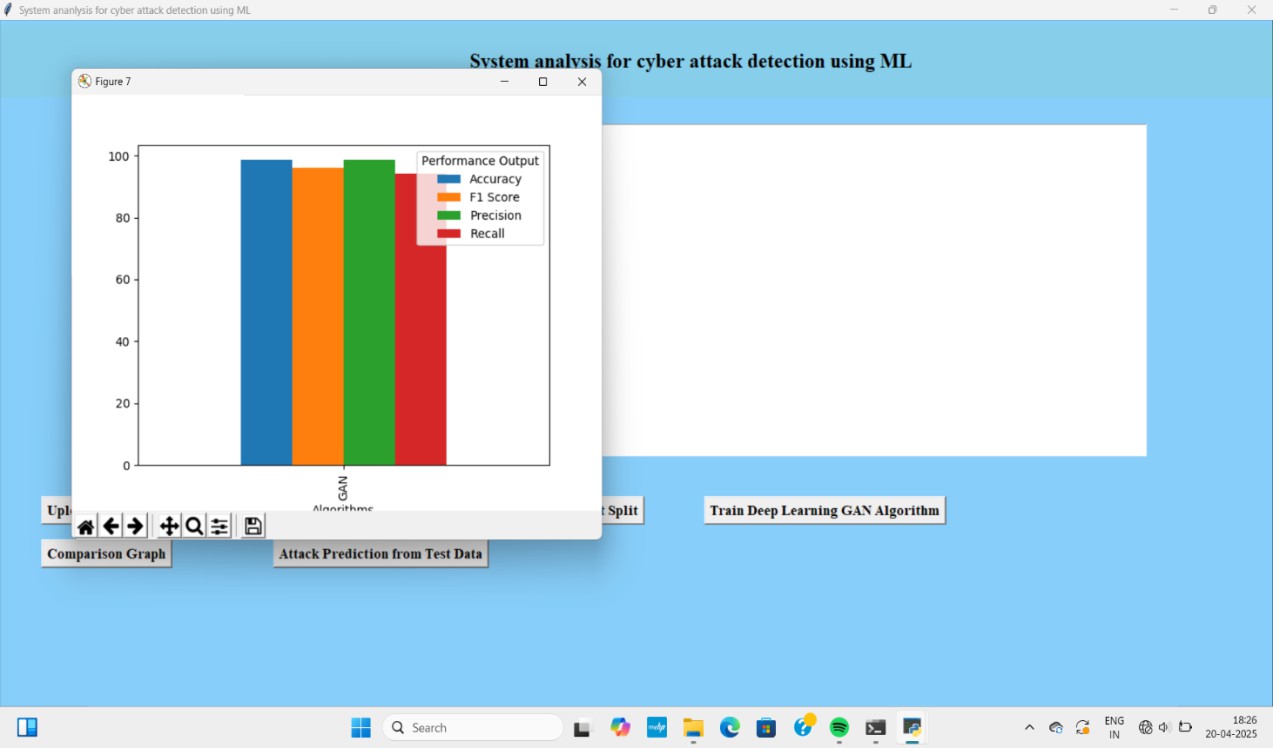
button to get below graph



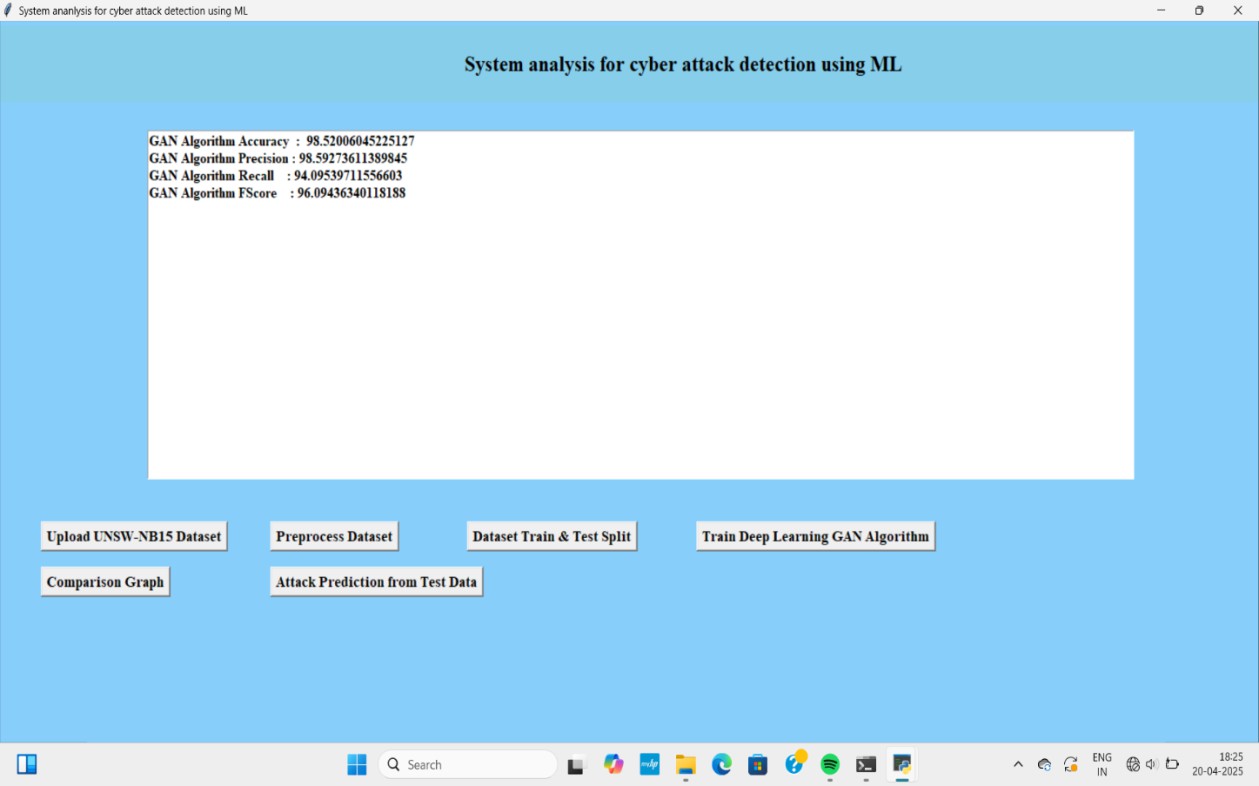
In above graph all different colour bars represents different metrics and can see all metrics are closer to 100%. Now close above graph and then click on ‘Attack Prediction from Test Data’ button to upload test data and get below output



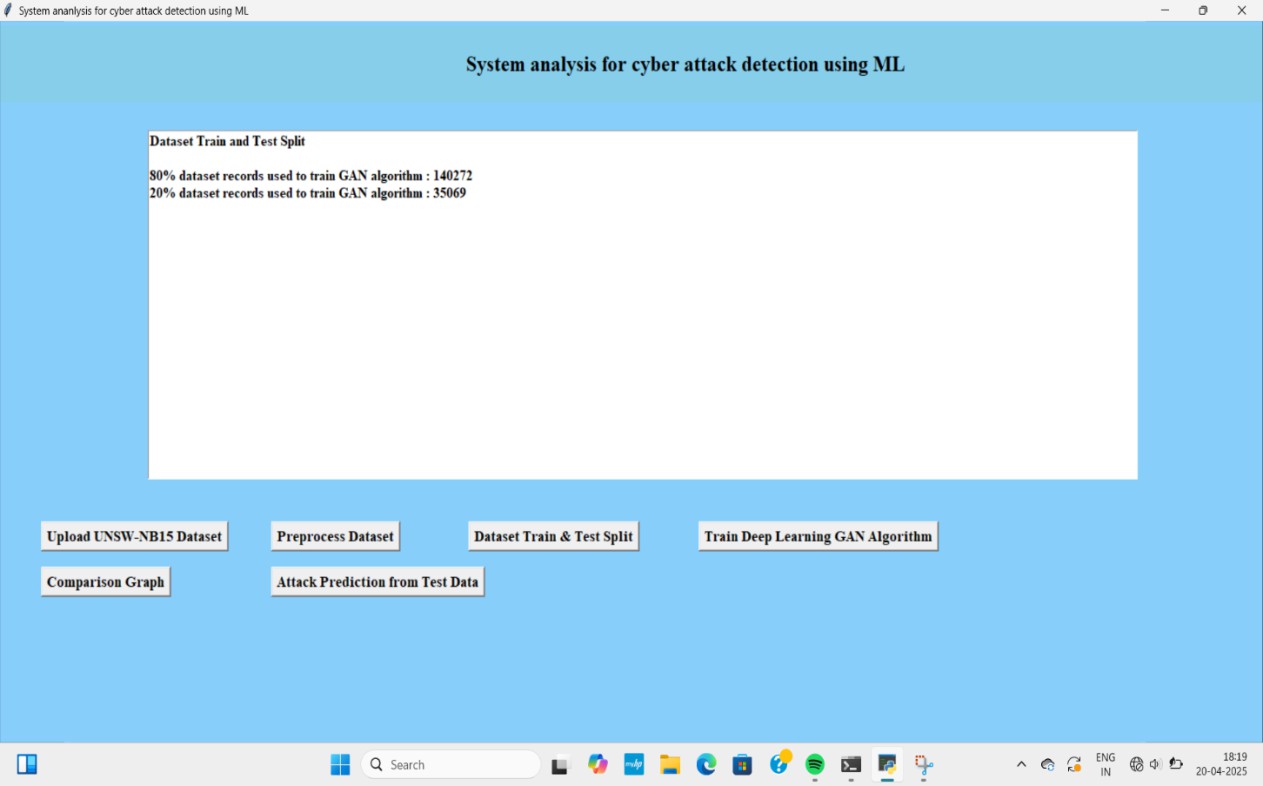
In above screen selecting and uploading ‘testdata.csv’ file and then click on ‘Open’ button to get below output



In above screen in square bracket can see Test data values of different signatures and after arrow = symbol can see predicted attack names



In above screen can see Brute Force and DOS attack



In above screen can see ‘Normal’ packet predicted.

Similarly by changing test data values you can perform prediction on normal or attack packets.

## CONCLUSION:

In conclusion, the application of machine learning in cyber attack detection offers a transformative approach to modern cybersecurity. Unlike traditional methods that rely on predefined rules and signatures, machine learning models can analyze vast amounts of data in real-time, identifying both known and novel threats. These models, by learning from continuous data streams, enhance the system’s ability to detect sophisticated attacks that may otherwise go unnoticed. With the ability to adapt to evolving threat landscapes, machine learning significantly improves the accuracy and efficiency of cyber attack detection, reducing the reliance on human intervention and offering proactive security solutions.

However, while machine learning shows great promise, challenges such as data imbalance, high false positive rates, and the need for frequent model updates remain. Overcoming these challenges is critical to realizing the full potential of machine learning in cybersecurity. Nevertheless, as technology advances and more sophisticated techniques emerge, machine learning will continue to play an essential role in shaping the future of cyber defense. By enhancing the scalability, adaptability, and effectiveness of detection systems, machine learning will help organizations stay one step ahead of increasingly sophisticated cyber threats, providing stronger protection against potential vulnerabilities.

## REFERENCES

* Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly Detection: A Survey. ACM Computing Surveys (CSUR), 41(3), 1-58.
* Sommer, R., & Paxson, V. (2010). Outside the Closed World: On Using Machine Learning for Network Intrusion Detection. IEEE Symposium on Security and Privacy.
* Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A Detailed Analysis of the KDD CUP 99 Data Set. IEEE Symposium on Computational Intelligence for Security and Defense Applications.
* Zhang, J., & Zulkernine, M. (2006). Anomaly Based Network Intrusion Detection with Unsupervised Outlier Detection. IEEE International Conference on Communications.
* Pektas, A., & Acarman, T. (2017). Application of Machine Learning Algorithms for Intrusion Detection. IEEE International Conference on Big Data and Smart Computing.
* Ring, M., Wunderlich, S., Scheuring, D., Landes, D., & Hotho, A. (2019). A Survey of Netwo
* rk-based Intrusion Detection Data Sets. Computers & Security, 86, 147-167.