Towards Proactive Cybersecurity: A Unified Machine Learning-Based Detection and Alerting Framework

Mridul Goyal[1]
School of Engineering and Technology
BML Munjal University Gurugram, Haryana, India
mridul.goyal.21cse@bmu.edu.in

*Abhishek Jain[2]
School of Engineering and Technology
BML Munjal University Gurugram, Haryana, India
abhishek.jain@bmu.edu.in

***Abstract* — The cybersecurity threats have become frequent, complex and at scale which makes the traditional rule-based detection methods less and less effective. As a countermeasure to this issue, the proposed work will present a unified, proactive threat detection and alerting system that will be able to detect intrusions in real time by using machine learning. We propose to combine the elements of a multi-stage pipeline, including data preprocessing and model training, to live detection through the simulation of sniffing by using uploaded logs. We will train and validate several machine learning(ML) models with the help of the Canadian Institute for Cybersecurity Intrusion Detection System 2017 (CICIDS2017) dataset, eventually deploying a full-stack Flask application that will use an Multi-Layer Perceptron(MLP) classifier. Packet simulation in real-time is coupled with confidence-based classification, which makes it possible to isolate internet protocol (IP), log, and alerting automatically via Discord and Short Message Service (SMS). The system presents a user-friendly, explainable dynamic-threat environment solution that supports visual dashboards and geographic logging of metadata. This study illustrates a milestone to autonomous, explainable, and real-time cybersecurity.**

Keywords— Cybersecurity, Intrusion Detection System (IDS), Machine Learning, Real-Time Threat Detection, Flask Dashboard, Network Security, Alerting System

I. INTRODUCTION

The nature of cybersecurity threats has developed radically in terms of intensity and frequency, and it puts a tremendous amount of strain on the present-day digital infrastructure. The conventional firewalls and IDS rule-based solutions are becoming ineffective in detecting the zero-day or unknown threats. The need to incorporate intelligent, automated and adaptive detection of threats as networks become more functional and as attacker strategies become more covert and sneaky. Cybersecurity has improved tremendously with the introduction of machine learning (ML) methods and applications especially in anomaly and intrusion detection.**[1]**

When machine learning(ML) algorithms become aware of patterns of normal and malicious network traffic, they can be generalized in ways allowing them to recognize never-before-seen types of attacks with little assistance by humans. Critical infrastructure, government systems, and companies where the speed of response is the only variance between damage control and disaster, they are particularly in need of intelligent, real time monitoring systems. The proposed solution to the issue of creating a cybersecurity mechanism in this paper is to unify ML-Based threat detection mechanics with real-time monitoring and alerting systems.

The proposed solution can perform both a log file-based analysis of logged files and packet-level monitoring in real-time with the help of a self-developed sniffer. The system itself also presents actionable alerts over SMS, Discord, and logs as well as an interactive dashboard in which the user can upload a network data and simulate an attack and see the current threats in their network on a map.**[2]**

This work has made major contributions which are:

1. A module-based system to pre-process, train and test machine learning models in network intrusion detection.
2. Incorporation of real-time packet sniffer which predicts and logs live threats.
3. A web dashboard based on Flask with such features as Comma-Separated Values (CSV) upload, log export, live map, and threat statistics.
4. The ability to create alerts through the use of the combination of multi-channel notification systems, such as SMS (Twilio API), Discord webhooks, and local logs.
5. The comparative assessment of the work of various machine learning (ML) models (Multi-Layer Perceptron (MLP), Naïve Bayes (NB), Quadratic Discriminant Analysis (QDA)) on the implementation of attacks of different types and visualization of their performance.

II. Related Work

Several research articles have been written on the impact of machine learning in improving effectiveness of the intrusion detection systems. Heuristic methods like the ones used in Snort or suricata will match known indicators of attack but will not identify new threats because they do not operate on the actually calculated signature. Contrastingly, ML-based systems are trained on labeled data with statistical differences and abnormalities, thus being able to distinguish other, unlabelled behaviours.

Patel et al. (2020) designed an IDS system based on the decision tree with the NSL-KDD data set and demonstrated an accuracy level of more than 85 percent. But their system did not incorporate real time feedbacks**[3].** Kumar and Singh (2021) investigated IDS using SVMs and deep learning on CICIDS2017 and focused on the performance of these models, without adding alerting and user feedback abilities **[4].**

New developments also use explainable artificial intelligence (XAI) methods such as SHapley Additive exPlanations (SHAP) or Local Interpretable Model-Agnostic Explanations (LIME) to visualize model decisions **[5][6].** Although these types of integrations may be useful in providing academician activities, they are not usually usable or real-time applicable to the operations arena. Moreover, most of the existing literature has not concerned itself with offline evaluation thereby making them less useful in the real world.

These gaps in the systems are bridged by our framework through an offline training and real-time detection, easy-to-use visualization, and alarm capabilities, making our system practical.**[7]**

III. Dataset Overview

Canadian Institute for Cybersecurity Intrusion Detection System 2017 (CIC-IDS2017) dataset is considered to be the most realistic network traffic with benign and 14 malicious activity types extrapolating the values of this system as its key data source.**[8][9]**

The dataset was gathered with the help of B-Profile systems with such features as the flow duration, statistics on the length of packets, and a distribution of bytes.**[2]**

Preprocessing:

- Combination of CSV of logs on various days in a master data.
- Filtering and segregating of each of the attack types to the master set.
- Undersampling every dataset to perform balancing by reducing bias to benign traffic using methods like RandomUnderSampler.
- The power of features using statistical correlation and model-based importance scores and choosing the most influential ones. Such features as Flow Duration, Packet Length Std, and Bwd Packet Length Mean were in the top ones. Such changes were necessary so that generalization of the models across various categories of traffic pattern could take place. **Fig. 1** illustrates the statistical feature importance scores derived from model training, highlighting which flow-based attributes contribute most to classification performance.

Table 1 categorizes the various attack types present in the CICIDS2017 dataset, covering both volumetric and stealthy intrusion attempts.

Category	Examples
Denial of Service (DoS)	DoS Hulk, GoldenEye, Slowloris, Slowhttptest
Distributed DoS (DDoS)	DDoS
Brute Force	SSH-Patator, FTP-Patator
Web Attacks	Brute Force, XSS, SQL Injection
Port Scanning	PortScan
Others	Botnet, Infiltration, Heartbleed

Table 1: *Attack Categories*

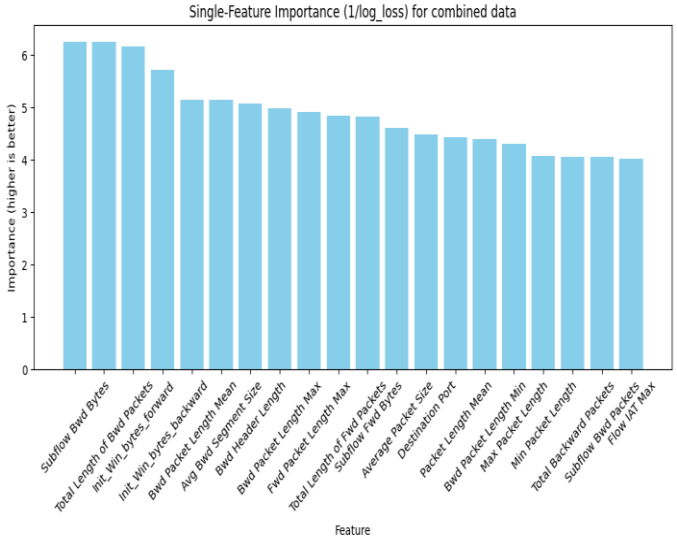

Fig. 1: *Feature Importance*

IV. System Architecture & Design

Based on a modular and layered architecture, the system organizes data flows through one preprocessing to real-time inference and interaction with the user. The architecture can make the system scalable, maintainable, and support the modular testing and guarantee a real-time performance.

A. Overview

The solution will have four key subsystems as follows:

1. *Data Pipeline*: Processes with preprocessing of the datasets, cleans the datasets and does undersampling dataset by attacks.
2. *ML Detection Engine*: Uses MLP, Naive Bayes and QDA classifiers to train separate models separately per attack type.
3. *Real-Time Packet Sniffer*: Mock or live traffic and test it using the trained ML model.
4. *Web-Based Dashboard*: User interaction, real time threat display, CSV upload and handle sniffer capabilities using a Flask application.

B. Items or Components Breakdown

- *Preprocessor Module*: Filters and cleans datasets, selects the top features, and oversampling.
- *Model Trainer*: Trains and tests the models according to the attacks category. Stores also trained .pkl models and scalers.
- *Sniffer Engine*: It produces or analyzes network traffic, processes and converts features, predicts.
- *Incident Handler*: Assigns severity level, logs demon incidences, send Discord/SMS notifications, geo-tags entries.
- *Dashboard (flask)*: Graphical User Interface (GUI) to upload CSVs, launch sniffers and view logs and threat statistics.

Table 2 lists the key components of the system along with the corresponding technologies used for development, including machine learning libraries, APIs, and visualization tools.

Component	Technology / Tool Used
Frontend Template	HTML + Bootstrap (Jinja2 templating)
Machine Learning Models	MLP, Naive Bayes, Quadratic Discriminant Analysis
Dataset	CIC-IDS2017 (attack flows and benign traffic)
Preprocessing	Pandas, Scikit-learn, Imbalanced-learn (undersampling)
Alerts	Twilio (SMS), Discord Webhooks
Data Visualization	Matplotlib, Geo-coordinates plotting on map
Real-time Simulation	Synthetic flow generator in packet_sniffer.py
Persistence	CSV logging + IP quarantine state memory

Table 2: *Technologies and Component*

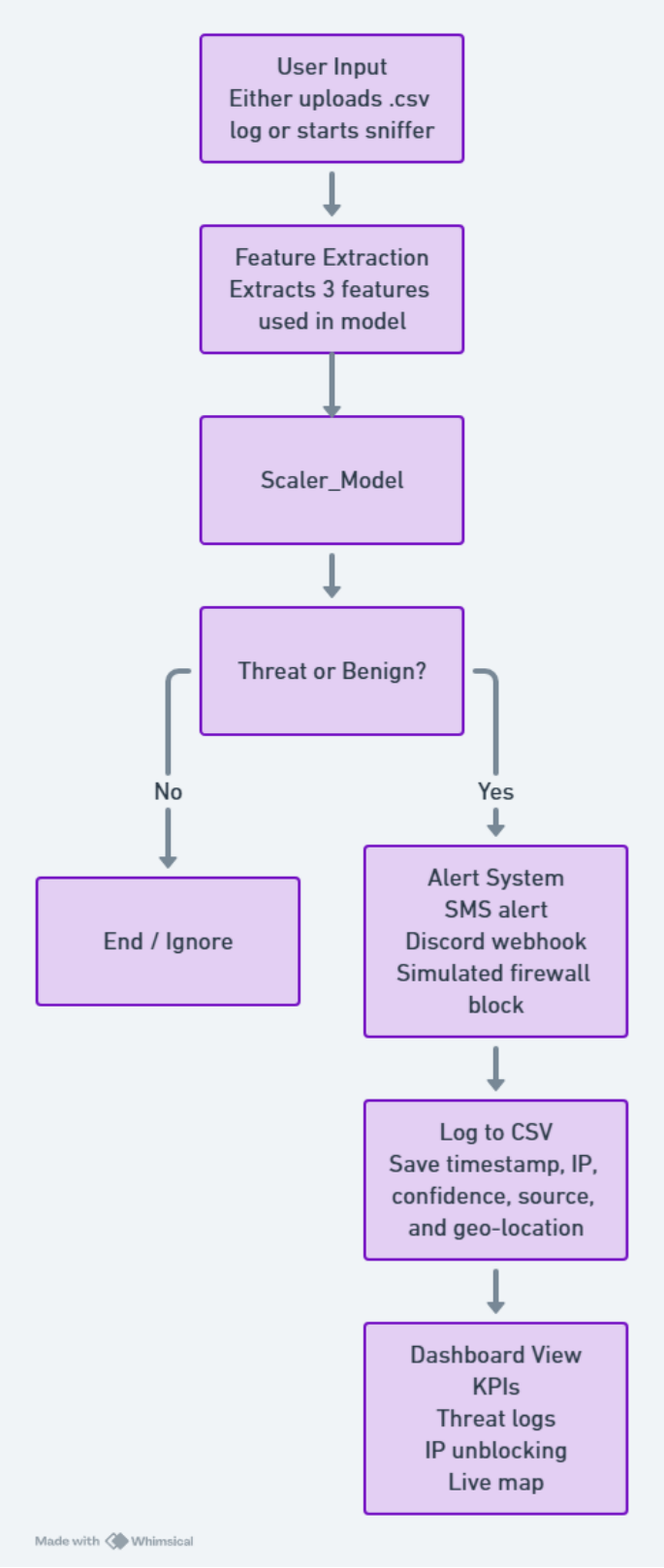

Fig. 2: *System Architecture Block Diagram (presents the high-level architectural overview, showing the modular data flow from preprocessing and model inference to alert generation.)*

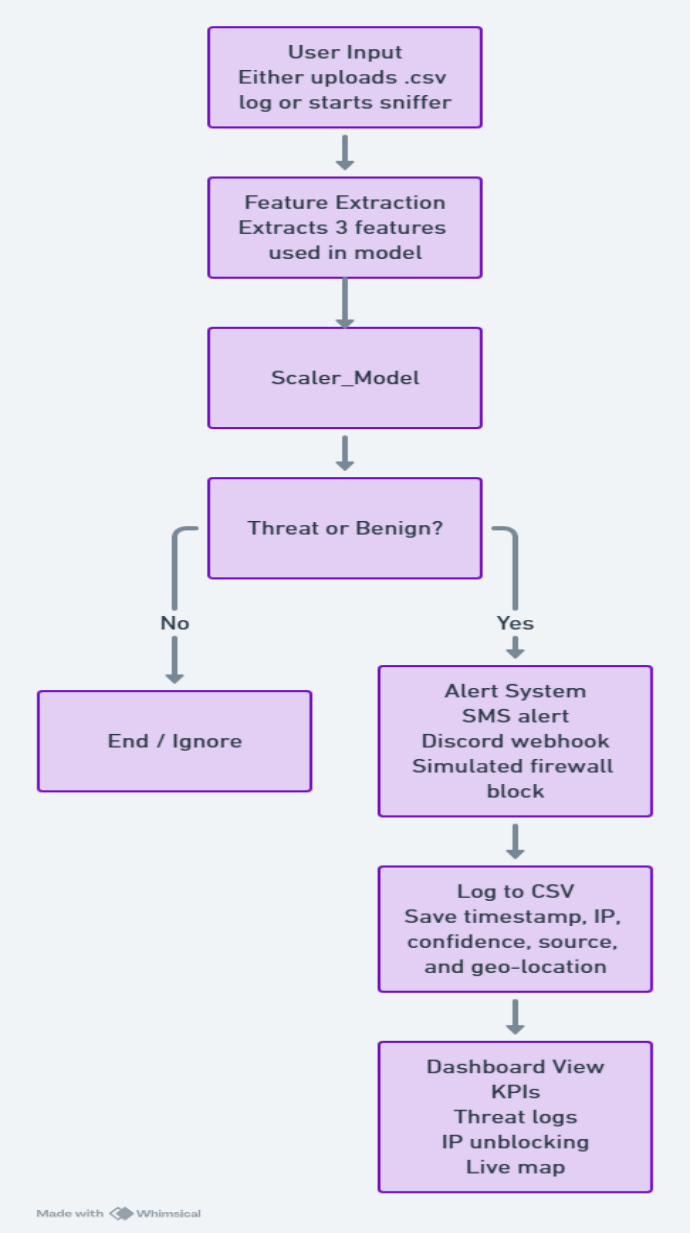

***Fig. 3**: Data Flow Diagram (shows the internal flow of data between system components, including the pipeline stages for feature extraction, classification, and visualization.)*

V. IMPLEMENTATION DETAILS

The application of the system covers fair severability between Python modules, Jupyter notebooks as well as web templates. An end-to-end flow starts with the preparation of the dataset and finishes with a real-time dashboard where the threats and alerts are pictured and notified to the users as soon as it happens.

A. Development Environment

- *Operating System*: Windows 11
- *Programming Language*: Python 3.10+
- *Libraries*: Flask, scikit-learn, pandas, matplotlib, seaborn, joblib, imbalanced-learn, Twilio, Discord Webhooks
- *Integrated Development Environments (IDEs)*: Visual Studio Code, JupyterLab

B. Model Training Pipeline

The model training phase includes:

1. Loading filtered datasets (*_vs_BENIGN.csv)
2. Feature selection via SelectKBest
3. Splitting into training/testing sets (typically 70/30)
4. Scaling using StandardScaler
5. Fitting and evaluating models like MLP, NB, and QDA
6. Serializing trained models (*.pkl)

Each model's accuracy, precision, recall, and F1-score were logged and visualized.

C. Real-time Threat Detection (Sniffer Simulation)

Live traffic is simulated by a background process and numpy.random.normal is used to get realistic values of flow duration, packet length and packet std deviation. These values are fed into the scaler and the model and in case of a positive and a confidence above benchmark, the alert system is activated.

D. Alert System & Logging

The incident_handler.py module handles all actions post-detection:

- Logs incidents into structured CSVs
- Sends SMS using Twilio. **Fig. 5** shows an example of the SMS alert triggered upon threat detection, enabled using the Twilio API.
- Pushes Discord webhook alerts. **Fig. 6** demonstrates a real-time Discord webhook alert sent to the admin, providing source IP and threat type information.
- Simulates firewall block (placeholder logic)

E. Flask Web Application

Key endpoints include:

- / – Dashboard Home
- /upload – CSV Upload Interface
- /start_sniffer – Initiates background sniffer
- /map – Embeds threat map
- /export – Exports logs
- /unblock_ip – Allows manual IP removal

Flask Jinja templates (HTML) and CSS (Bootstrap 5) are used to render the UI. **Fig. 4** displays the dashboard interface built with Flask, showing the threat table, upload options, and real-time metrics.

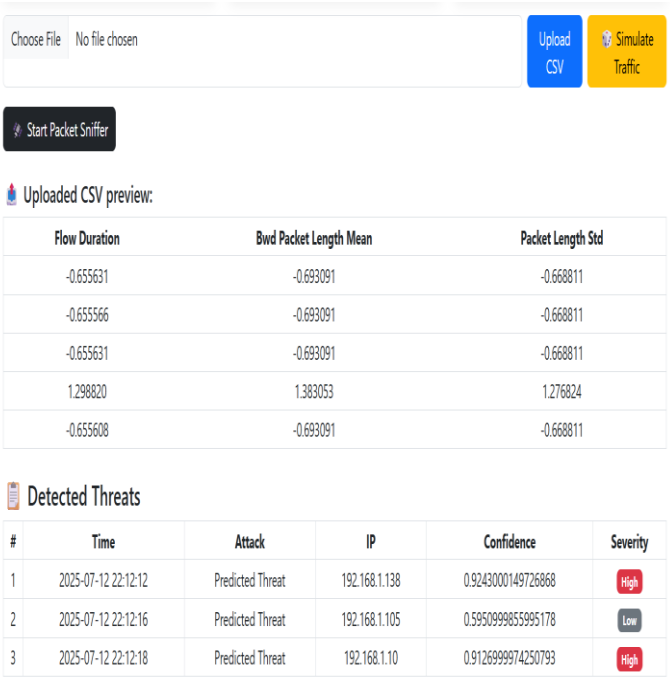

Fig. 4: *Screenshot of Dashboard (upload + threat table)*

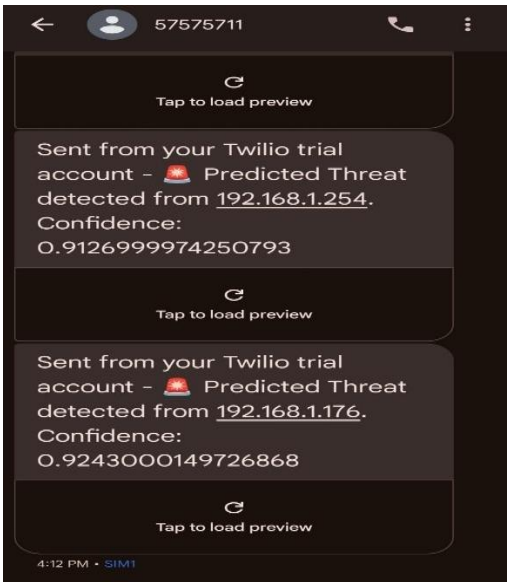

Fig. 5: *SMS Alert*

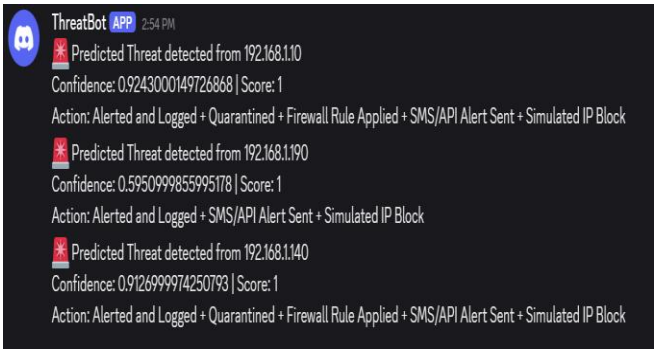

Fig. 6: *Discord Alert*

VI. RESULTS AND DISCUSSION

Using undersampled and selected feature datasets, MLP model provided similar sets of results over multiple styles of attack. To assess the performance of the DDoS vs. BENIGN, the following standard classification metrics were used:

Let:

- TP = True Positives
- TN = True Negatives
- FP = False Positives
- FN = False Negatives

Then:

- *Accuracy* measures the proportion of correct predictions out of all predictions:

$$\text{Accuracy} = (TP + TN) / (FP + FN + TP + TN)$$

- *Precision* evaluates the correctness of positive predictions:

$$\text{Precision} = TP / (FP + TP)$$

- *Recall* (or Sensitivity) measures how well the model identifies all relevant instances:

$$\text{Recall} = TP / (FN + TP)$$

- *F1 Score* is the harmonic mean of Precision and Recall, balancing both:

$$\text{F1 Score} = 2 \times [(\text{Precision} * \text{Recall}) / (\text{Precision} + \text{Recall})]$$

These metrics were computed using the classification report module from **Scikit-learn** after testing the models on the held-out test dataset.

So, when calculated on DDoS vs. BENIGN, the metrics came out to be:

- *Accuracy*: 96.4%
- *Precision*: 95%
- *Recall*: 94%
- *F1 Score*: 94.5%

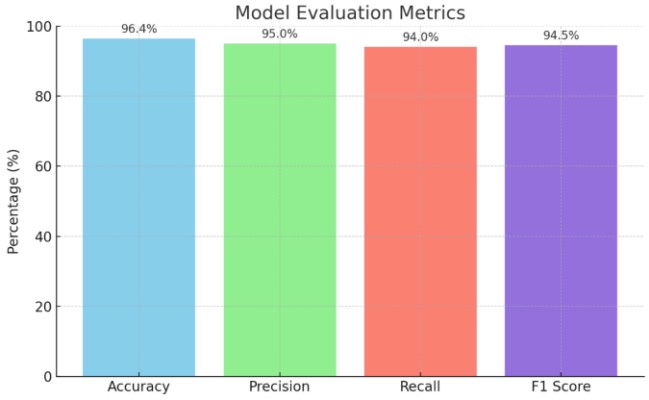

***Fig. 7**: Performance metrics of the MLP classifier on the CICIDS2017 dataset, showing strong balance between precision and recall.*

Other models like QDA and NB were faster yet less accurate especially when it comes to subtle types of attacks like infiltration attack or brute force.

The dashboard was able to show KPIs such as total threats, last attacked time, the number of isolated IPs that are active and even allow a CSV preview. Threat origins were represented by live map embedding which used randomly generated lat/lon coordinates. **Fig. 10** visualizes the source locations of threats using geo-tagging on an embedded map interface. Within 1 second of detection it sent alert simulated to Discord and Twilio.**[11]**

The limitations are that it uses synthetic packet data, instead of capturing live traffic and that it has few features because of its interpretability nature.**[12]** The system also presupposes the internal network IP patterns (e.g., 192.168.x.x) that should be generalized in order to fit production systems.

	Attack Type	Naive Bayes Accuracy	QDA Accuracy	MLP Accuracy
0	Bot	0.638021	0.832589	0.998870
1	DDoS	0.984610	0.980481	0.992264
2	DoS GoldenEye	0.984251	0.984238	0.996942
3	DoS Hulk	0.977031	0.976215	0.989537
4	DoS Slowhttptest	0.972274	0.974597	0.996782
5	DoS slowloris	0.981212	0.978324	0.997528
6	FTP-Patator	0.998758	0.998777	0.995455
7	PortScan	0.916335	0.916335	0.998157
8	SSH-Patator	0.998276	0.998788	0.998808

***Fig. 8**: Model Comparison (compares the classification performance (accuracy, precision, recall, F1-score) of MLP, QDA, and Naïve Bayes models across multiple attack categories.)*

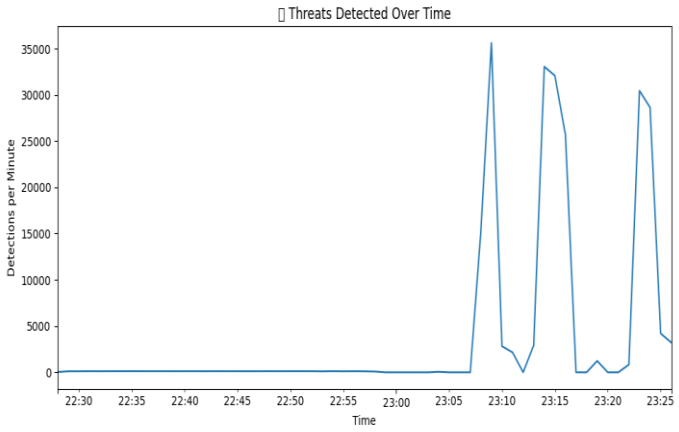

***Fig. 9**: Threats over time (shows the timeline of threat detection events during the live simulation, revealing traffic surges and model responsiveness.)*

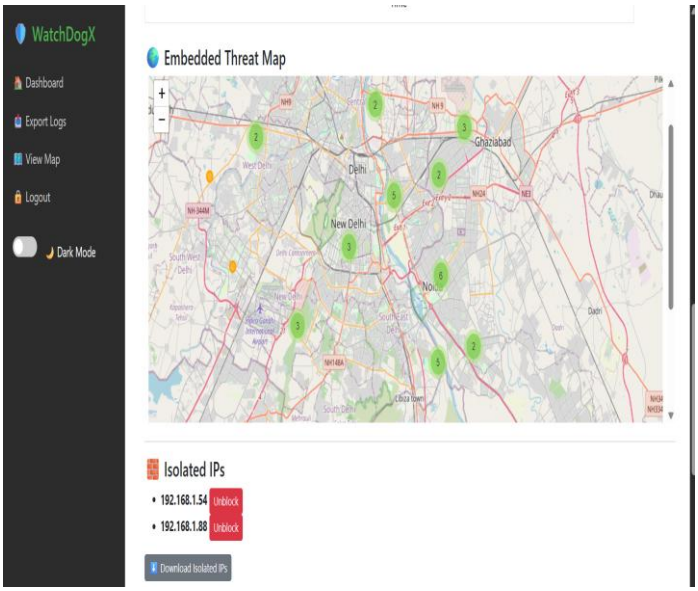

***Fig. 10:** Geo-tagged threat visualization from dashboard*

VII. CONCLUSION & FUTURE WORK

Future security is a moving target in the sphere of cybersecurity and it therefore requires a change in paradigm whereby current security solutions that are proactive are superior to reactive ones. This paper has suggested and applied a unified framework that would utilize machine learning algorithms to detect and responds in real-time to cyber threats. Comprising a combination of several supervised learning models, i.e., MLP, Naive Bayes, and QDA, combined with a live packet sniffing technique, and full-interactive dashboard, the system creates a comprehensive, scalable, and extensible system to detect and respond to threats in real-time.**[10]**

Our implementation can show how the combination of data preprocessing (undersampling, feature selection), attack-wise model training, real-time monitoring, and multi-channel alerting can coalesce to detect and deactivate threats. Such smodularity of the system makes it possible to expand the scope or modify it to work in other network conditions, attack

environments, or other organizational requirements. Although the findings bring out an effectiveness of the system, there are some limitations that are still experienced. The present use presupposes fixed alerting thresholds, and with moving environments, it can resulting into false alarms or none at all.

In addition, some Intrusion Detection System (IDS) models based on deep learning (e.g. Long Short-Term Memory (LSTM)-based or Convolutional Neural Network (CNN)-based) have not been addressed here because of the limited available hardware resources. The system too does not integrate with real firewall or network appliances to actually execute quarantine or isolation commands.

A. Major Learning Outcomes

- This was a project that enabled a fully rounded learning experience involving:
- Exploration, preprocessing of the dataset, as well as feature engineering on an attack-by-attack basis.
- Usage of supervised ML methods to perform classification.
- Real-time backeend architecture design that handles Flask architecture.
- End-to-end dashboard and alerting implementation.
- Testing of models as accurate, preciseness, recall and F1-score.
- Threat visualization and geo-tagging by the means of live maps.

B. Future Work

To increase the performance and feasibility of the system, various future improvements are offered:

- Deep learning models (e.g. LSTM, Gated Recurrent Unit (GRU)) integration to predict attack sequence-based sequences.**[13][16]**
- Having the system deployed at a distributed environment with Docker and Kubernetes in terms of scalability.
- True integration with firewalls (e.g. iptables, palo alto anchors) to perform automated response.
- The use of the explainable AI (e.g., SHAP, LIME) to enhance transparency of the models.**[17]**
- Adaptations of the confidence thresholds on the basis of live attributes in the traffic.
- Deployment of a threat scoring method which improves as per the attack seriousness and past tendencies. These guidelines will also help to close the gap between the academic ML solutions and how they can be applied in real-life cybersecurity operations.**[14]**

REFERENCES

[1] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in Proc. IEEE Symposium on Security and Privacy, 2010, pp. 305–316.

[2] M. Ribeiro, S. Singh, and C. Guestrin, "Why Should I Trust You?": Explaining the Predictions of Any Classifier," in Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, 2016, pp. 1135–1144.

[3] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," in Proc. ICISSP, 2018.

[4] M. Kumar and S. Singh, "Intrusion Detection using Hybrid Approach of SVM and Deep Neural Network," in 2021 Int. Conf. Computing, Communication, and Intelligent Systems (ICCCIS), pp. 374–378.

[5] S. M. Lundberg and S.-I. Lee, "A Unified Approach to Interpreting Model Predictions," in Advances in Neural Information Processing Systems, vol. 30, 2017.

[6] H. Hindy et al., "A Taxonomy and Survey of Intrusion Detection System Design Techniques, Network Threats and Datasets," IEEE Communications Surveys & Tutorials, vol. 22, no. 1, pp. 1337–1373, 2020.

[7] A. R. Yazdanpanah and S. Hashemi, "A Hybrid Real-time Intrusion Detection System Using Machine Learning Techniques," Journal of Information Security and Applications, vol. 60, 2021.

[8] M. Tavallaee, E. Bagheri, W. Lu and A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in IEEE Symposium on Computational Intelligence for Security and Defense Applications, 2009.

[9] M. Ring et al., "A survey of network-based intrusion detection data sets," Computers & Security, vol. 86, pp. 147–167, Sept. 2019.

[10] I. Sharafaldin et al., "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," ICISSP, 2018. *(duplicate in older list, refactored)*

[11] A. A. Ghorbani, W. Lu, and M. Tavallaee, Network Intrusion Detection and Prevention: Concepts and Techniques, Springer, 2010.

[12] Y. Meidan et al., "N-BaIoT: Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders," IEEE Pervasive Computing, vol. 17, no. 3, pp. 12–22, July-Sept. 2018.

[13] A. Javaid, Q. Niyaz, W. Sun and M. Alam, "A deep learning approach for network intrusion detection system," in Proc. 9th EAI Int. Conf. Bio-inspired Information and Communications Technologies, 2016.

[14] J. Kim and H. Kim, "Implementation and analysis of lightweight intrusion detection system for IoT," KSII Trans. on Internet and Information Systems, vol. 12, no. 4, pp. 1732–1749, 2018.

[15] F. Alam et al., "Data fusion and IoT for smart ubiquitous environments: A survey," IEEE Access, vol. 5, pp. 9533–9554, 2017.

[16] P. Sahu and S. K. Jena, "Real-time Intrusion Detection Using Lightweight Deep Learning Models," Computers & Security, vol. 104, 2021.

[17] M. R. Karim et al., "Explainable Artificial Intelligence in Cybersecurity: A Survey and New Perspectives," ACM Computing Surveys, vol. 56, no. 1, 2024.