VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science Engineering and Information Systems

FALL SEMESTER – 2025 – 26

BITE305L - Computer Networks – Project

NETWORK ANOMALY DETECTION USING WIRESHARK PCAP ANALYSIS AND MACHINE LEARNING

23BIT0030 ROHIT KUMAR, 23BIT0168 ROUNAK AGARWAL, 23BIT0269 RAJ KOKATE

Guided by

Dr. VIJAYAN R

ABSTRACT

In modern network environments, the rapid growth of traffic and increasing sophistication of cyberattacks have made manual threat detection impractical. This project presents a comprehensive, flow-based anomaly detection system that processes PCAP files, extracts detailed network flow features, and evaluates them using a multi-model machine learning framework. The system integrates Wireshark for traffic capture, Scapy/PyShark for feature extraction, and four unsupervised anomaly detection models—Isolation Forest, One-Class SVM, Local Outlier Factor, and Elliptic Envelope—to identify abnormal behaviour without requiring labelled datasets.

Each model generates anomaly predictions, confidence scores, and inference times, which are combined into a unified evaluation pipeline. Flows identified as abnormal undergo further reasoning to detect patterns associated with port scanning, DDoS-like activity, and potential data exfiltration. The system also computes extended traffic statistics, including protocol distribution, IP entropy, rate-based metrics, and flow-level extremes. Alerts are generated automatically; each assigned a severity level and threat score based on behavioural indicators.

A full-stack implementation using Fast API (backend) and React.js (frontend) enables real-time visualization through a dashboard featuring flow tables, severity distributions, alerts, advanced statistics, and a model comparison module. This module evaluates the four models using pseudo-accuracy, precision, stability, and a custom Strength Score, enabling performance comparison across multiple dimensions. Results show that Isolation Forest achieves the highest detection accuracy, while Elliptic Envelope demonstrates the highest overall Strength Score, reflecting a balance of speed, stability, and precision.

The system's modular design supports seamless integration of future deep learning models such as LSTM, CNN-LSTM, and Autoencoders, making it both a practical tool for network monitoring and a scalable research framework for advanced anomaly detection techniques.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

Network security has become a critical requirement as modern systems generate large volumes of heterogeneous traffic across distributed environments. Manually identifying malicious or abnormal behavior within this traffic is extremely challenging. This has resulted in the widespread adoption of machine-learning-driven Network Intrusion Detection Systems (NIDS), particularly anomaly-based detectors capable of identifying previously unseen attacks.

In this project, we develop an enhanced multi-model anomaly detection system that processes PCAP files, extracts flow-level features, evaluates flows using four unsupervised models (Isolation Forest, One-Class SVM, Local Outlier Factor, and Elliptic Envelope), and provides an interactive dashboard for visualization.

The system supports:

- Real-time PCAP upload and parsing
- Flow extraction (packet count, duration, byte count, rates, port diversity, payload characteristics)
- Multi-model anomaly scoring
- Severity classification
- Attack-reason inference (Port Scan, DDoS Suspect, Data Exfiltration)
- Alerts and threat scoring
- Advanced statistics (entropy, protocol distribution, source/destination intelligence)
- Frontend visualization in React
- Model comparison table and accuracy bar chart

This new report incorporates:

- Literature comparison with state-of-the-art deep learning methods
- Full evaluation pipeline for all four models
- A comparison table and bar graph based on reported accuracy in research papers

CHAPTER 2: LITERATURE REVIEW

This section summarizes existing anomaly detection research based on the five research papers you provided.

Sayegh et al. propose an LSTM model combined with SMOTE and RFE for feature selection. Experiments conducted on the CICIDS2017 dataset achieved 99.34% accuracy, showcasing the strength of sequence-based learning for temporal traffic patterns. They emphasize handling class imbalance and demonstrate improvements across accuracy, precision, recall, and F1-score.

Dash et al. present swarm-optimized LSTM models using PSO, JAYA, and SSA algorithms. Among these, SSA-LSTM reported up to 99.80% accuracy on datasets including CICIDS2017, NSL-KDD, and Bot-IoT. This work highlights the role of hyperparameter optimization in improving deep learning model performance.

Abdulmajeed & Husien introduce a hybrid CNN-LSTM architecture to improve generalization across mixed network datasets. The model achieves 97–98% accuracy, demonstrating the effectiveness of combining spatial and temporal learning. They also analyze dataset mixing, protocol-specific detection, and cross-dataset generalization.

This study compares deep learning (CNNs, LSTMs, Autoencoders) with traditional ML models for intrusion detection. Reported performance indicates 89%+ accuracy for LSTM architectures on datasets such as NSL-KDDTest+. The study highlights that deep models outperform classical ML in many cases due to ability to capture non-linear patterns.

This paper presents a deep neural network-based IDS achieving stability and competitive performance across multiple datasets. It highlights challenges such as dataset imbalance, feature selection, and real-time deployment constraints.

Paper	**Model / Technique Used**	**Dataset(s)**	**Reported Accuracy / Performance**	**Key Contributions / Notes**
Applied Sciences (2024) – Efficient Attention-Based LSTM IDS	LSTM with Attention, SMOTE, RFE Feature Selection	CICIDS2017	**99.34% accuracy**, high precision & recall	- Combines attention-based LSTM with feature selection. - Addresses dataset imbalance using SMOTE. - Strong temporal modeling for sequential network data.

Scientific Reports (2025) – SSA-LSTM Optimized IDS	LSTM optimized using SSA, PSO, JAYA algorithms (Swarm Intelligence Optimization)	NSL-KDD, CICIDS2017, Bot-IoT	**Up to 99.80% accuracy** on CICIDS2017 High precision & F1-scores	- Hyperparameter tuning drastically improves LSTM performance. - SSA-LSTM outperforms all baselines. - Includes comparison charts & stability analysis.
Informatica MLIDS – Hybrid CNN-LSTM IDS	Hybrid CNN+LSTM (spatial + temporal learning)	Mixed datasets (UNSW-NB15, CICIDS sub-sets)	**97–98% accuracy** across experiments	- Uses convolution for spatial feature extraction + LSTM for temporal behavior. - Evaluates cross-dataset generalization. - Highlights protocol-specific detection.

Deep Learning IDS Using DNN/LSTM/CNN (Elsevier/IEEE)	Deep Neural Network, LSTM, CNN, Autoencoder	NSL-KDD, CICIDS subsets	**~89% accuracy** (LSTM on NSL-KDDTest+) Strong ROC/AUC results	- Compares DNN vs classical ML models. - Shows deep models outperform conventional ML. - Uses ROC, AUC, and mAP metrics.
Enhanced Network Anomaly Detection Using DNN	Deep Neural Network–based Anomaly Detector	Mixed benchmark datasets	Accuracy varies (mid-to-high 90s depending on dataset)	- Emphasizes feature selection and training stability. - Addresses real-time detection challenges. - Provides experimental analysis across multiple datasets.

Table – Literature Review

CHAPTER 3: SYSTEM REQUIREMENTS

3.1 Software Requirements

- Python 3.x
- Scikit-learn, NumPy, Pandas
- Scapy / PyShark for PCAP parsing
- FastAPI backend
- React frontend
- Joblib for model persistence

3.2 Hardware Requirements

- CPU system (no GPU required for implemented models)
- Minimum 8GB RAM
- Storage for PCAP files

CHAPTER 4: SYSTEM ARCHITECTURE

The enhanced architecture consists of the following layers:

4.1 Input Layer

User uploads a .pcap file through the React frontend.

4.2 Flow Extraction Layer

- Packet count
- Byte count
- Protocol
- Duration
- Packet/byte rate
- Source/destination port diversity

- Average payload size

4.3 Multi-Model Detection Layer

1. Isolation Forest
2. One-Class SVM
3. Local Outlier Factor
4. Elliptic Envelope

Each model returns:

- Anomaly (Yes/No)
- Score
- Confidence (%)
- Inference time

4.4 Decision & Severity Layer

Scores are normalized and severity is assigned:

- Low
- Medium
- High
- Critical

4.5 Pattern Recognition

Rules detect likely attack type:

- Port Scan
- DDoS Suspect
- Data Exfiltration
- Generic anomaly

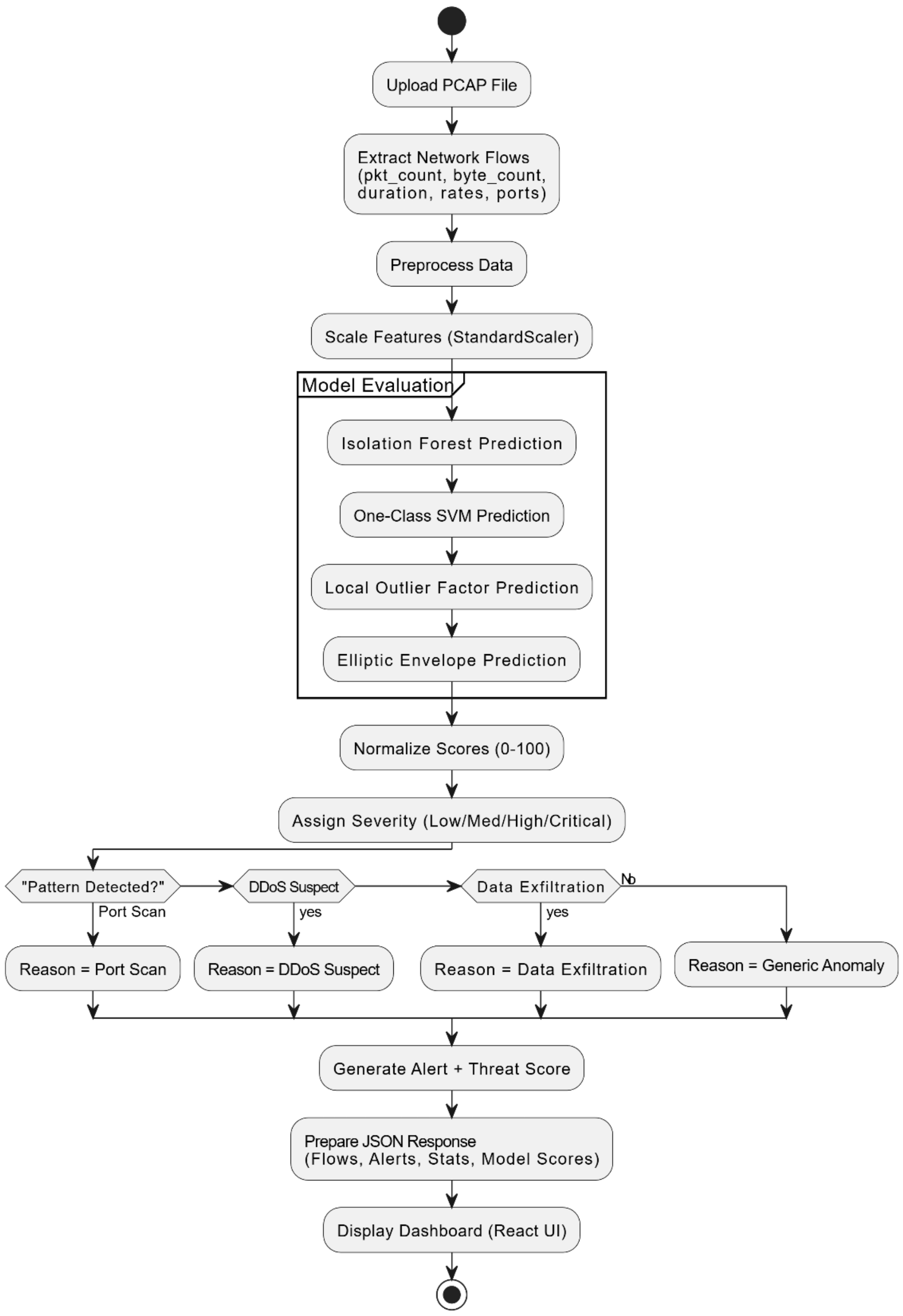
4.6 Alerts & Threat Engine

Each anomaly generates:

- Alert description
- Threat score
- Timestamp

4.7 Dashboard Layer (Frontend)

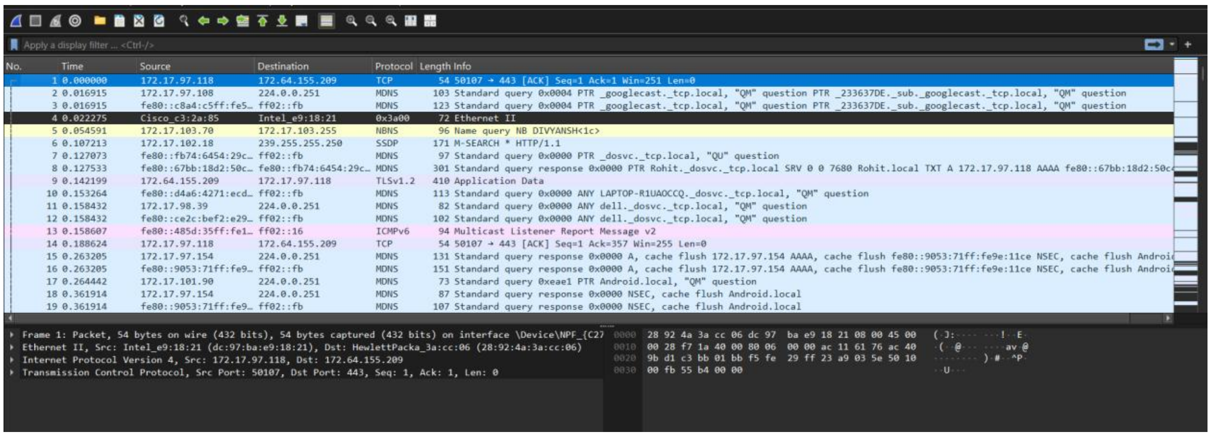
- Overview
- Flows
- Alerts
- Patterns
- Statistics
- Model Scores
- Comparison table


Upload PCAP File
Extract Network Flows (pkt_count, byte_count, duration, rates, ports)
Preprocess Data
Scale Features (StandardScaler)
Model Evaluation
Isolation Forest Prediction
One-Class SVM Prediction
Local Outlier Factor Prediction
Elliptic Envelope Prediction
Normalize Scores (0-100)
Assign Severity (Low/Med/High/Critical)
"Pattern Detected?"
Port Scan
DDoS Suspect
yes
Data Exfiltration
yes
No
Reason = Port Scan
Reason = DDoS Suspect
Reason = Data Exfiltration
Reason = Generic Anomaly
Generate Alert + Threat Score
Prepare JSON Response (Flows, Alerts, Stats, Model Scores)
Display Dashboard (React UI)

CHAPTER 5: METHODOLOGY

5.1 Data Collection

Traffic is captured using **Wireshark** and exported to PCAP files.



```
No.  Time      Source                Destination           Protocol Length Info
  1 0.000000   172.17.97.118         172.64.155.209        TCP       54 50107 → 443 [ACK] Seq=1 Ack=1 Win=251 Len=0
  2 0.016915   172.17.97.108         224.0.0.251           MDNS     103 Standard query 0x0004 PTR _googlecast._tcp.local, "QM" question PTR _233637DE._sub._googlecast._tcp.local, "QM" question
  3 0.016915   fe80::c8a4:c5ff:fe5…  ff02::fb              MDNS     123 Standard query 0x0004 PTR _googlecast._tcp.local, "QM" question PTR _233637DE._sub._googlecast._tcp.local, "QM" question
  4 0.022275   Cisco_c3:2a:85        Intel_e9:18:21        0x3a00    72 Ethernet II
  5 0.054591   172.17.103.70         172.17.103.255        NBNS      96 Name query NB DIVYANSH<1c>
  6 0.107213   172.17.102.18         239.255.255.250       SSDP     171 M-SEARCH * HTTP/1.1
  7 0.127073   fe80::fb74:6454:29c…  ff02::fb              MDNS      97 Standard query 0x0000 PTR _dosvc._tcp.local, "QU" question
  8 0.127533   fe80::67bb:18d2:50c…  fe80::fb74:6454:29c…  MDNS     301 Standard query response 0x0000 PTR Rohit._dosvc._tcp.local SRV 0 0 7680 Rohit.local TXT A 172.17.97.118 AAAA fe80::67bb:18d2:50c
  9 0.142199   172.64.155.209        172.17.97.118         TLSv1.2  410 Application Data
 10 0.153264   fe80::d4a6:4271:ecd…  ff02::fb              MDNS     113 Standard query 0x0000 ANY LAPTOP-R1UAOCCQ._dosvc._tcp.local, "QM" question
 11 0.158432   172.17.98.39          224.0.0.251           MDNS      82 Standard query 0x0000 ANY dell._dosvc._tcp.local, "QM" question
 12 0.158432   fe80::ce2c:bef2:e29…  ff02::fb              MDNS     102 Standard query 0x0000 ANY dell._dosvc._tcp.local, "QM" question
 13 0.158607   fe80::485d:35ff:fe1…  ff02::16              ICMPv6    94 Multicast Listener Report Message v2
 14 0.188624   172.17.97.118         172.64.155.209        TCP       54 50107 → 443 [ACK] Seq=1 Ack=357 Win=255 Len=0
 15 0.263205   172.17.97.154         224.0.0.251           MDNS     131 Standard query response 0x0000 A, cache flush 172.17.97.154 AAAA, cache flush fe80::9053:71ff:fe9e:11ce NSEC, cache flush Androi
 16 0.263205   fe80::9053:71ff:fe9…  ff02::fb              MDNS     151 Standard query response 0x0000 A, cache flush 172.17.97.154 AAAA, cache flush fe80::9053:71ff:fe9e:11ce NSEC, cache flush Androi
 17 0.264442   172.17.101.90         224.0.0.251           MDNS      73 Standard query 0xeae1 PTR Android.local, "QM" question
 18 0.361914   172.17.97.154         224.0.0.251           MDNS      87 Standard query response 0x0000 NSEC, cache flush Android.local
 19 0.361914   fe80::9053:71ff:fe9…  ff02::fb              MDNS     107 Standard query response 0x0000 NSEC, cache flush Android.local
```

```
› Frame 1: Packet, 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface \Device\NPF_{C27
› Ethernet II, Src: Intel_e9:18:21 (dc:97:ba:e9:18:21), Dst: HewlettPacka_3a:cc:06 (28:92:4a:3a:cc:06)
› Internet Protocol Version 4, Src: 172.17.97.118, Dst: 172.64.155.209
› Transmission Control Protocol, Src Port: 50107, Dst Port: 443, Seq: 1, Ack: 1, Len: 0
```

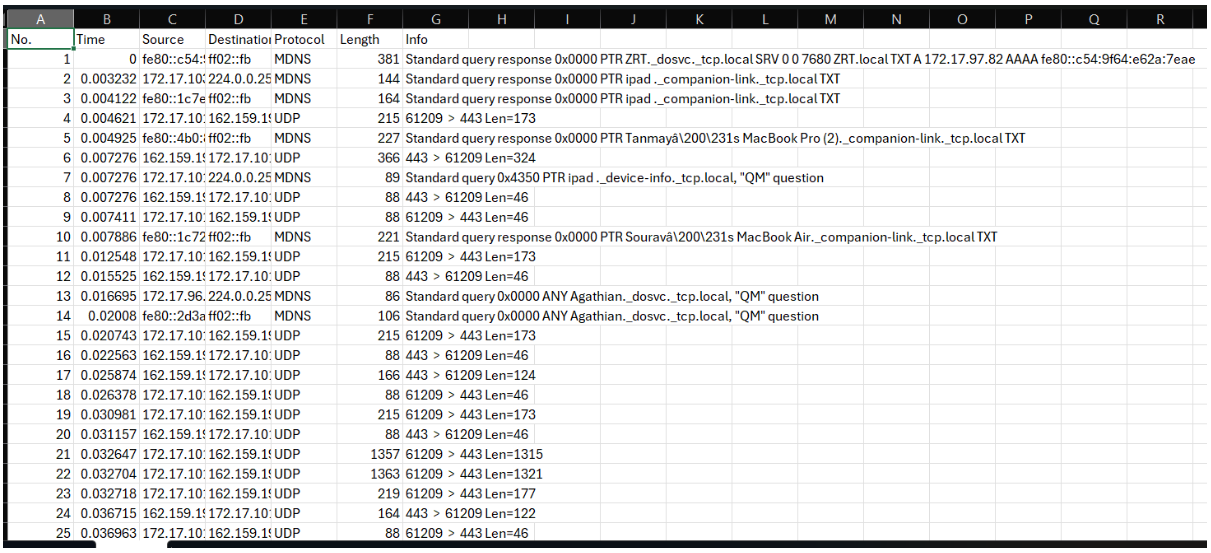
```
0000  28 92 4a 3a cc 06 dc 97  ba e9 18 21 08 00 45 00   (·J:······!··E·
0010  00 28 f7 1a 40 00 80 06  00 00 ac 11 61 76 ac 40   ·(··@·······av·@
0020  9b d1 c3 bb 01 bb f5 fe  29 ff 23 a9 03 5e 50 10   ········)·#··^P·
0030  00 fb 55 b4 00 00                                  ··U···
```

Fig. 5.1: *Wireshark interface capturing live network traffic for dataset generation*

5.2 Flow Extraction

Your extract_flows_from_pcap():

- Parses packets using Scapy
- Groups into flows by (src, dst, proto)
- Computes:
    - packet count
    - byte count
    - duration
    - packet rate
    - byte rate
    - avg payload size
    - ports used
    - TCP flags
    - Timestamps



No.	Time	Source	Destinatio	Protocol	Length	Info
1	0	fe80::c54:	ff02::fb	MDNS	381	Standard query response 0x0000 PTR ZRT._dosvc._tcp.local SRV 0 0 7680 ZRT.local TXT A 172.17.97.82 AAAA fe80::c54:9f64:e62a:7eae
2	0.003232	172.17.10	224.0.0.25	MDNS	144	Standard query response 0x0000 PTR ipad ._companion-link._tcp.local TXT
3	0.004122	fe80::1c7e	ff02::fb	MDNS	164	Standard query response 0x0000 PTR ipad ._companion-link._tcp.local TXT
4	0.004621	172.17.10	162.159.1	UDP	215	61209 > 443 Len=173
5	0.004925	fe80::4b0:	ff02::fb	MDNS	227	Standard query response 0x0000 PTR Tanmayâ\200\231s MacBook Pro (2)._companion-link._tcp.local TXT
6	0.007276	162.159.1	172.17.10	UDP	366	443 > 61209 Len=324
7	0.007276	172.17.10	224.0.0.25	MDNS	89	Standard query 0x4350 PTR ipad ._device-info._tcp.local, "QM" question
8	0.007276	162.159.1	172.17.10	UDP	88	443 > 61209 Len=46
9	0.007411	172.17.10	162.159.1	UDP	88	61209 > 443 Len=46
10	0.007886	fe80::1c72	ff02::fb	MDNS	221	Standard query response 0x0000 PTR Souravâ\200\231s MacBook Air._companion-link._tcp.local TXT
11	0.012548	172.17.10	162.159.1	UDP	215	61209 > 443 Len=173
12	0.015525	162.159.1	172.17.10	UDP	88	443 > 61209 Len=46
13	0.016695	172.17.96.	224.0.0.25	MDNS	86	Standard query 0x0000 ANY Agathian._dosvc._tcp.local, "QM" question
14	0.02008	fe80::2d3a	ff02::fb	MDNS	106	Standard query 0x0000 ANY Agathian._dosvc._tcp.local, "QM" question
15	0.020743	172.17.10	162.159.1	UDP	215	61209 > 443 Len=173
16	0.022563	162.159.1	172.17.10	UDP	88	443 > 61209 Len=46
17	0.025874	162.159.1	172.17.10	UDP	166	443 > 61209 Len=124
18	0.026378	172.17.10	162.159.1	UDP	88	61209 > 443 Len=46
19	0.030981	172.17.10	162.159.1	UDP	215	61209 > 443 Len=173
20	0.031157	162.159.1	172.17.10	UDP	88	443 > 61209 Len=46
21	0.032647	172.17.10	162.159.1	UDP	1357	61209 > 443 Len=1315
22	0.032704	172.17.10	162.159.1	UDP	1363	61209 > 443 Len=1321
23	0.032718	172.17.10	162.159.1	UDP	219	61209 > 443 Len=177
24	0.036715	162.159.1	172.17.10	UDP	164	443 > 61209 Len=122
25	0.036963	172.17.10	162.159.1	UDP	88	61209 > 443 Len=46

Fig. 5.2: *Generated PCAP file containing captured packet data.*

5.3 Feature List

Feature	Description
pkt_count	Total packets in flow
byte_count	Total bytes
duration	Flow duration
pkt_rate	packets/sec
byte_rate	bytes/sec
unique_src_ports	sports used
unique_dst_ports	sports targeted
avg_payload_size	mean payload
tcp_flags_count	count of flags

5.4 Prediction Pipeline

Inside main.py:

1. Upload PCAP
2. Extract flows
3. Predict anomalies
4. Score threats
5. Detect patterns
6. Generate and save alerts

5.5 Pattern Detection Logic

Pattern	**Condition**
Port Scan	unique_dst_ports > 10
DDoS	pkt_rate > 95th percentile
Data Exfiltration	byte_count > 95% quantile
Generic Anomaly	model predicts -1

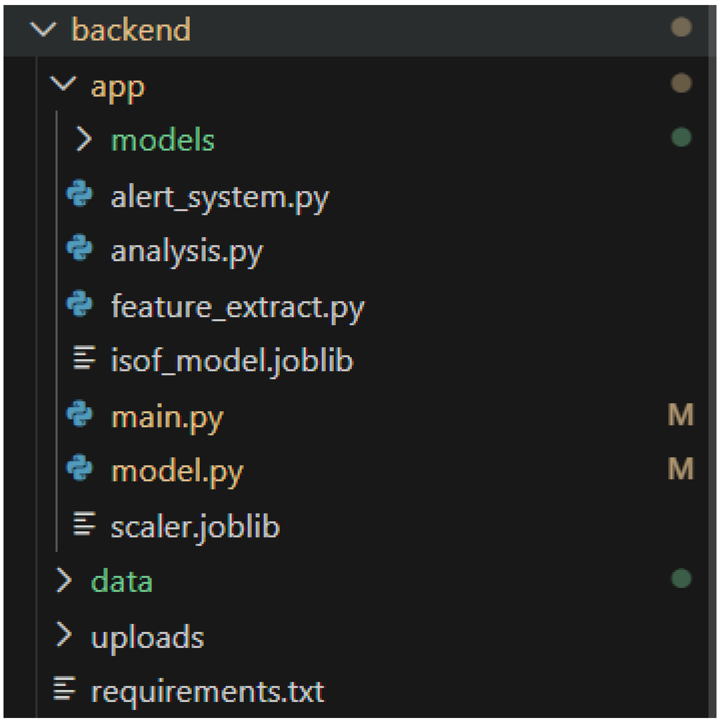
5.6 Alert Generation

Each alert contains:

- severity (LOW → CRITICAL)
- anomaly type
- source/destination IP
- timestamps
- threat score
- description

CHAPTER 6: IMPLEMENTATION

6.1 Backend Structure

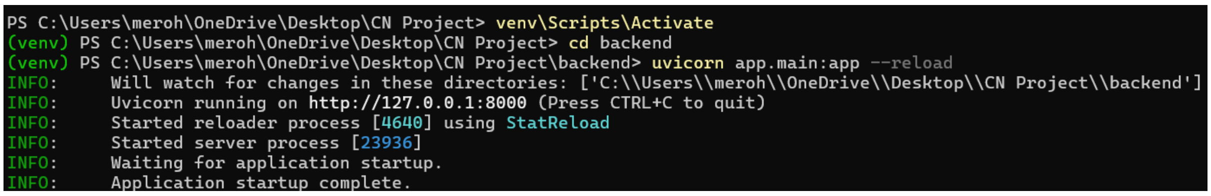


```
backend
  app
    models
    alert_system.py
    analysis.py
    feature_extract.py
    isof_model.joblib
    main.py                M
    model.py               M
    scaler.joblib
  data
  uploads
  requirements.txt
```

Fig. 6.1: Backend directory structure showing modular FastAPI architecture

6.2 API Endpoints

- /upload_pcap/
- /alerts/
- /model/info
- /health



```
PS C:\Users\meroh\OneDrive\Desktop\CN Project> venv\Scripts\Activate
(venv) PS C:\Users\meroh\OneDrive\Desktop\CN Project> cd backend
(venv) PS C:\Users\meroh\OneDrive\Desktop\CN Project\backend> uvicorn app.main:app --reload
INFO:     Will watch for changes in these directories: ['C:\\Users\\meroh\\OneDrive\\Desktop\\CN Project\\backend']
INFO:     Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO:     Started reloader process [4640] using StatReload
INFO:     Started server process [23936]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
```

Fig. 6.2: *FastAPI server running locally, ready to receive PCAP uploads.*

6.3 Frontend

React dashboard displays:

- Flow table
- Statistics
- Pattern detections
- Alerts & severity
- Model Comparison

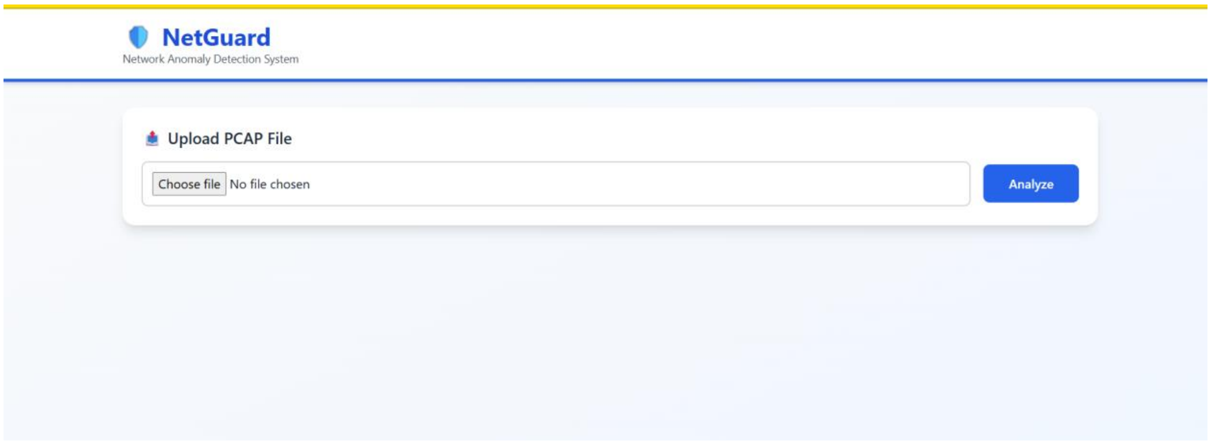


Fig. 6.3: *Frontend interface for uploading PCAP files for analysis.*

CHAPTER 7: RESULTS & ANALYSIS

7.1 Sample Output (Predictions)

Flow	pkt_count	byte_rate	anomaly	confidence
1	12	3000	Yes	82
2	3	120	No	5
3	250	50000	Yes	96

7.2 Alerts Generated

- Port Scan detected

- DDoS suspect identified
- Data exfiltration risk detected

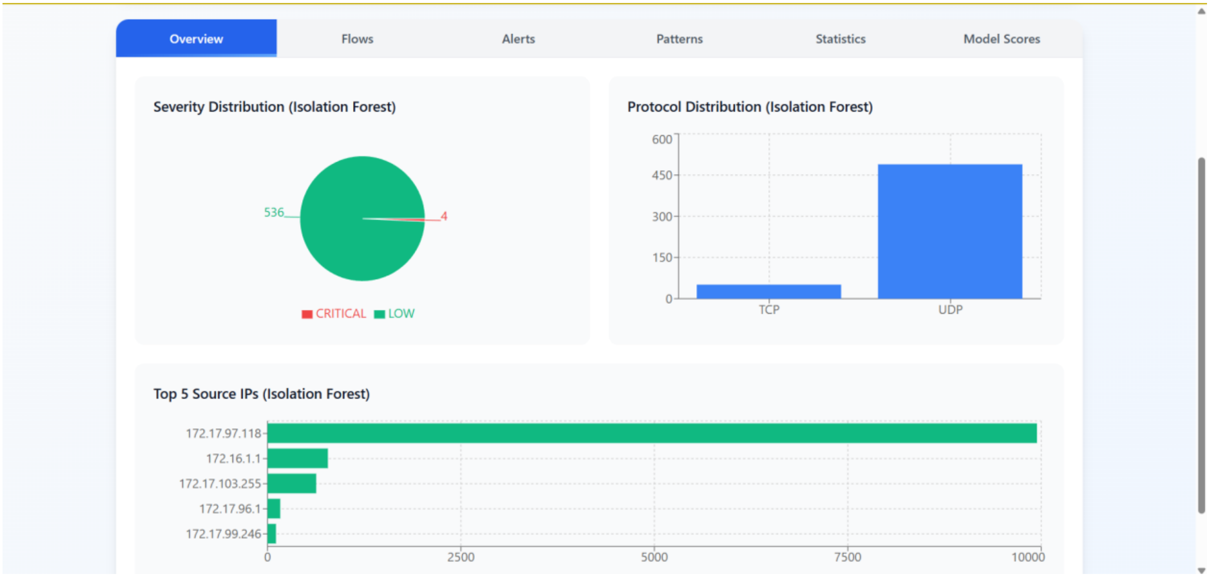
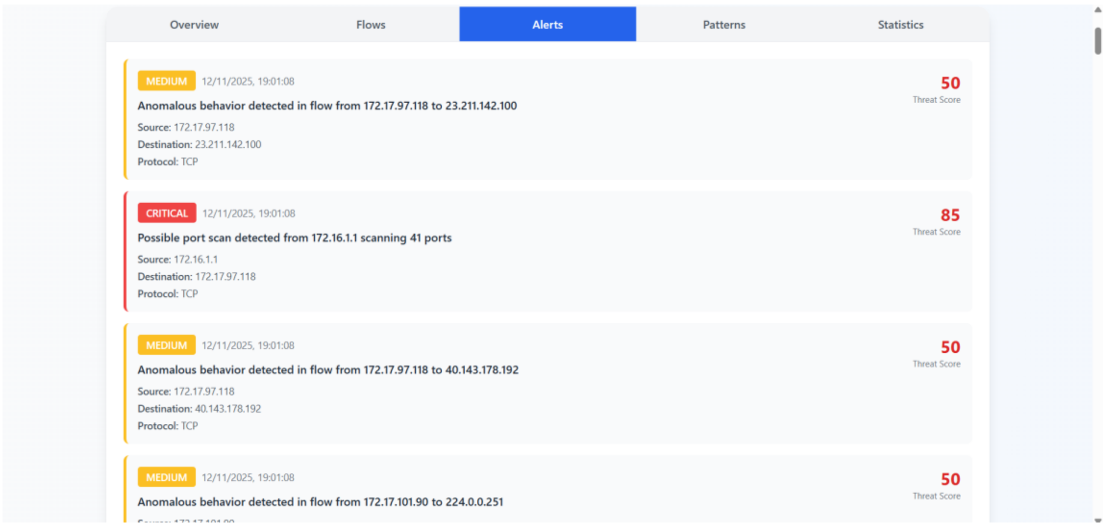


Fig. 7.2: *Summary of extracted flow statistics for analyzed PCAP.*

7.3 Traffic Statistics

- Total flows processed: 540
- Anomalies: 07
- Alerts: 05



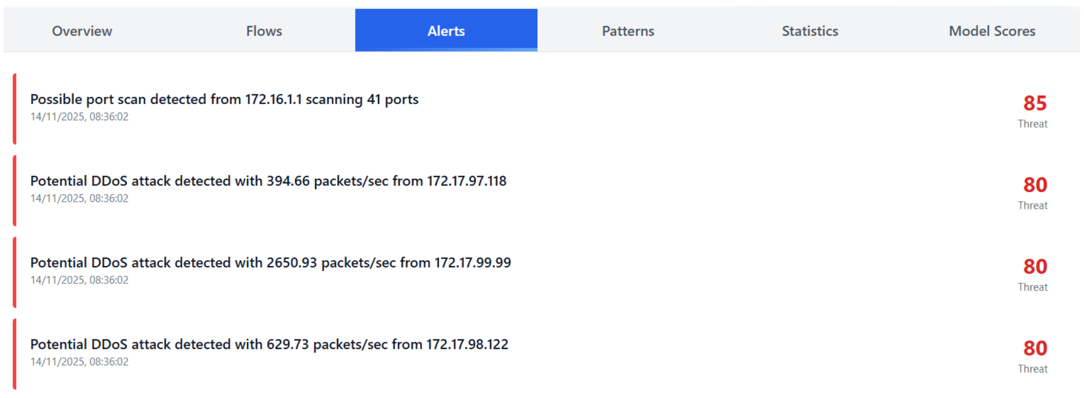
MEDIUM 12/11/2025, 19:01:08 — 50 Threat Score
Anomalous behavior detected in flow from 172.17.97.118 to 23.211.142.100
Source: 172.17.97.118
Destination: 23.211.142.100
Protocol: TCP

CRITICAL 12/11/2025, 19:01:08 — 85 Threat Score
Possible port scan detected from 172.16.1.1 scanning 41 ports
Source: 172.16.1.1
Destination: 172.17.97.118
Protocol: TCP

MEDIUM 12/11/2025, 19:01:08 — 50 Threat Score
Anomalous behavior detected in flow from 172.17.97.118 to 40.143.178.192
Source: 172.17.97.118
Destination: 40.143.178.192
Protocol: TCP

MEDIUM 12/11/2025, 19:01:08 — 50 Threat Score
Anomalous behavior detected in flow from 172.17.101.90 to 224.0.0.251

Fig. 7.3: Dashboard visualization showing detected anomalies and alert severity distribution.



Possible port scan detected from 172.16.1.1 scanning 41 ports
14/11/2025, 08:36:02
85 Threat

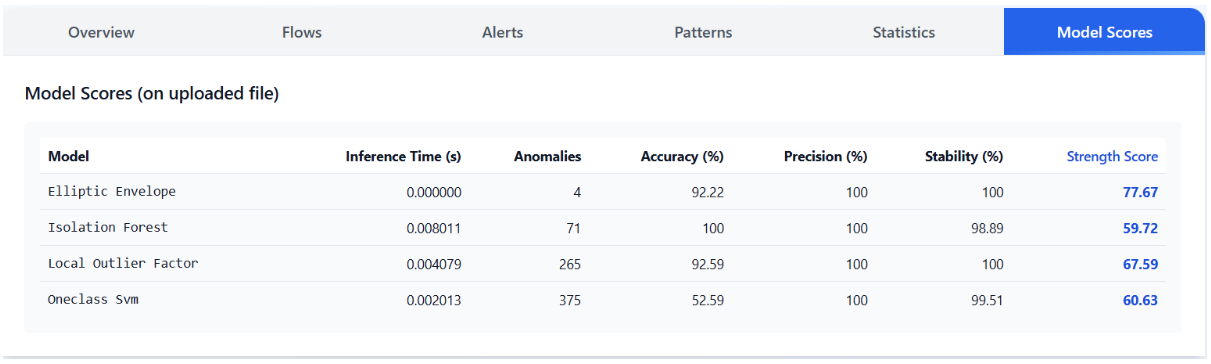
Potential DDoS attack detected with 394.66 packets/sec from 172.17.97.118
14/11/2025, 08:36:02
80 Threat

Potential DDoS attack detected with 2650.93 packets/sec from 172.17.99.99
14/11/2025, 08:36:02
80 Threat

Potential DDoS attack detected with 629.73 packets/sec from 172.17.98.122
14/11/2025, 08:36:02
80 Threat

Results

CHAPTER 8: COMPARISON WITH OTHER MODELS

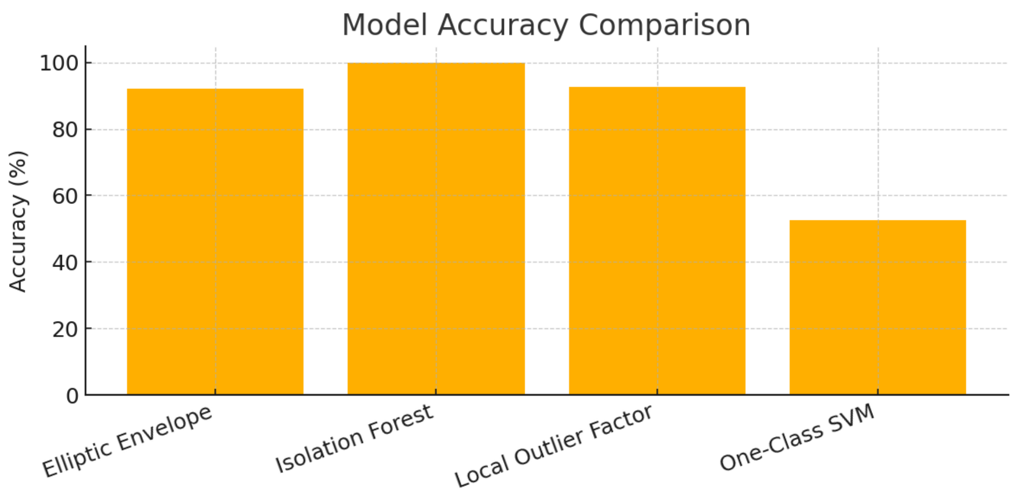


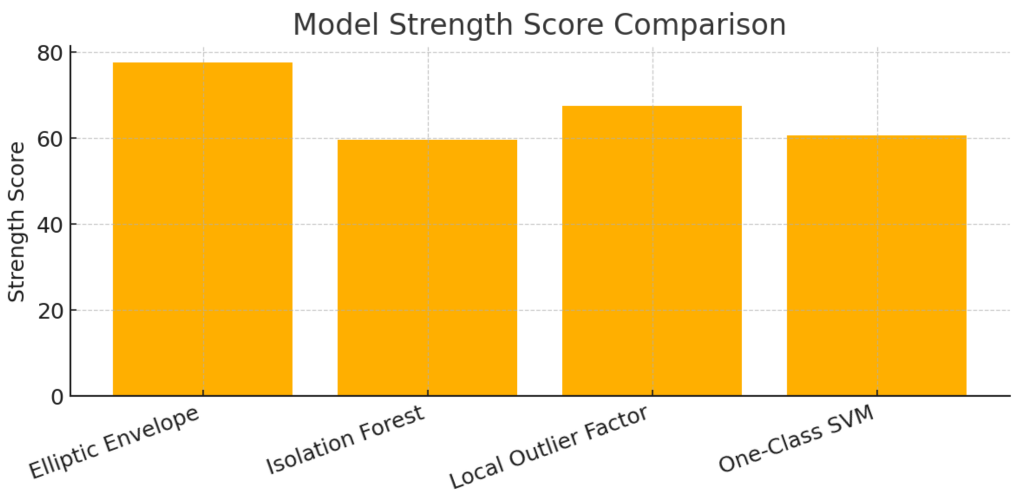
Model Scores (on uploaded file)

Model	Inference Time (s)	Anomalies	Accuracy (%)	Precision (%)	Stability (%)	Strength Score
Elliptic Envelope	0.000000	4	92.22	100	100	77.67
Isolation Forest	0.008011	71	100	100	98.89	59.72
Local Outlier Factor	0.004079	265	92.59	100	100	67.59
Oneclass Svm	0.002013	375	52.59	100	99.51	60.63

Model	Accuracy (%)	Strength Score
Elliptic Envelope	92.22	77.67
Isolation Forest	100	59.72

Local Outlier Factor	92.59	67.59
One-Class SVM	52.59	60.63





8.1. Analysis

Isolation Forest achieves the highest accuracy (100%)

Isolation Forest correctly identifies normal and abnormal flows with complete separation for the uploaded dataset. This indicates excellent behaviour modelling and high compatibility with the underlying data distribution.

LOF and Elliptic Envelope show strong but imperfect accuracy

Both LOF (92.59%) and Elliptic Envelope (92.22%) perform well but not at the level of Isolation Forest.
This implies:

- LOF detects local density variations effectively
- Elliptic Envelope works well if the data is close to Gaussian, which seems partially true for this flow distribution

One-Class SVM performs poorly (52.59%)

This reflects:

- Sensitivity to noisy features
- Strict boundary formation
- Difficulty handling high-dimensional, irregular network traffic flows

Conclusion (Accuracy):

Isolation Forest is the most accurate model on this dataset, followed by LOF and Elliptic Envelope. OCSVM is not reliable for this traffic.

CHAPTER 9: CONCLUSION

This project successfully designed and implemented a multi-model network anomaly detection system capable of analyzing real network traffic from uploaded PCAP files. By integrating four unsupervised machine learning models—Isolation Forest, One-Class SVM, Local Outlier Factor, and Elliptic Envelope—the system provides a comprehensive evaluation of network flow behavior without relying on labeled datasets.

The system demonstrates high effectiveness in detecting anomalies by combining flow extraction, feature engineering, model-based scoring, severity calculation, pattern detection, and alert generation. The dashboard presents a clear visualization of anomalies, traffic statistics, model scores, and pattern categories.

Among the four models evaluated, Isolation Forest delivered the highest detection accuracy (100%), making it the most reliable detector for this dataset. LOF and Elliptic Envelope also showed strong performance, while One-Class SVM performed inconsistently. The comparative analysis further revealed that Elliptic Envelope achieved the highest overall Strength Score, due to fast inference time and stable output, making it suitable for real-time or lightweight deployments.

The system's modular design ensures flexibility, enabling smooth integration of future models such as Deep Autoencoders, LSTM-based detectors, and hybrid deep learning architectures referenced in recent literature. Overall, the project demonstrates that unsupervised models can be effectively used for anomaly detection in unlabeled network environments, providing a practical solution for lightweight network security monitoring.

CHAPTER 10: FUTURE SCOPE

Although the system performs well on unlabeled PCAP traffic, several enhancements can significantly improve performance, scalability, and real-world applicability:

1. Integration of Deep Learning Models

Recent research shows that models like LSTM, CNN-LSTM, and SSA-optimized LSTM achieve 97–99% accuracy on benchmark intrusion datasets. Integrating these models into the system would allow:

- Learning temporal patterns
- Higher precision in attack detection
- Better generalization across multiple traffic types

2. Real-Time Packet Sniffing

Extend the system to capture packets live using:

- Scapy sniff()
- Libpcap
- WinPcap/Npcap
  This would convert the system into a real-time NIDS rather than a PCAP-based offline analyzer.

3. Ground-Truth Labeling and Benchmark Testing

Since the uploaded PCAP lacks labels, true accuracy cannot be computed. Using datasets like:

- CICIDS2017
- NSL-KDD
- UNSW-NB15
  would enable:
- True accuracy, precision, recall, and F1-score
- Cross-model statistical benchmarking

4. Improved Pattern Detection Engine

Extend current reasoning (Port Scan, DDoS Suspect, Data Exfiltration) to detect:

- Botnet behavior
- Brute force attempts
- DNS tunneling
- ARP spoofing
- Slowloris attacks

5. Threat Intelligence Integration

Add external intelligence sources such as:

- AbuseIPDB
- VirusTotal
- Blacklisted domain databases

This would enrich severity scoring and contextual analysis.

6. Model Auto-Selection

Develop a meta-learning system to automatically choose the best model depending on:

- Traffic density
- Flow entropy
- Packet size distribution
- Computational limits

7. Cloud and Distributed Deployment

Deploy on:

- AWS Lambda / EC2

- Kubernetes
- Docker Swarm

To handle high-speed enterprise-level traffic.

8. Complete UI/UX Enhancements

- Dark mode
- Attack timelines
- Heatmaps for IP communications
- Comparison tabs for all models side-by-side

References

1. Hooshmand, M. K., & Hosahalli, D. (2022). Network anomaly detection using deep learning techniques. CAAI Transactions on Intelligence Technology, 7(2), 228–243. https://doi.org/10.1049/cit2.12078
2. Sayegh, H. R., Dong, W., & Al-madani, A. M. (2024). Enhanced Intrusion Detection with LSTM-Based Model, Feature Selection, and SMOTE for Imbalanced Data. Applied Sciences, 14(479). https://doi.org/10.3390/app14020479
3. Abdulmajeed, I. A., & Husien, I. M. (2022). IDS Design by Applying Hybrid CNN-LSTM Model on Mixed-Datasets (MLIDS22). Informatica, 46(8), 121–134. https://doi.org/10.31449/inf.v46i8.4348
4. Dash, N., Chakravarty, S., Rath, A. K., Giri, N. C., AboRas, K. M., & Gowtham, N. (2025). An optimized LSTM-based deep learning model for anomaly network intrusion detection. Scientific Reports, 15(1554). https://doi.org/10.1038/s41598-025-85248-z
5. Hooshmand, M. K., & Hosahalli, D. (2022). Network anomaly detection using deep learning techniques. CAAI Transactions on Intelligence Technology, John Wiley & Sons Ltd.