PROJECT REPORT ON

AI-ML-for-Networking

A Toolkit for Intelligent Traffic Analysis & Anomaly Detection

Submitted By

Sambu Abhinaya Sree (237Y1A6202)

Annam Lakshman (237Y1A6223)

PV. Raghunath (237Y1A6236)

COMPUTER SCIENCE AND ENGINEERING (Cyber Security)

Marri Laxman Reddy Institute of Technology and Management

Under the Guidance of

Dr. Venkata Reddy (Associate Professor)

Submitted to

Intel® Unnati Program

Supported by

Edgate Technologies

In partnership with

Marri Laxman Reddy Institute of Technology and Management (MLRITM)

Date of Submission: July 2025

INDEX

FIGURES

Problem Statement

Description:

Modern networks face increasing challenges in monitoring and securing traffic due to the exponential growth of data, encrypted communication, and sophisticated cyber threats. Traditional rule-based security measures and deep packet inspection (DPI) techniques are becoming less effective in detecting and classifying threats, especially in encrypted traffic. Manual intervention in network traffic classification is inefficient, leading to delayed threat detection and security vulnerabilities. To address these issues, AI-driven solutions can analyze traffic patterns, detect anomalies, classify applications, and enhance security in real-time, ensuring adaptive and intelligent network defense.

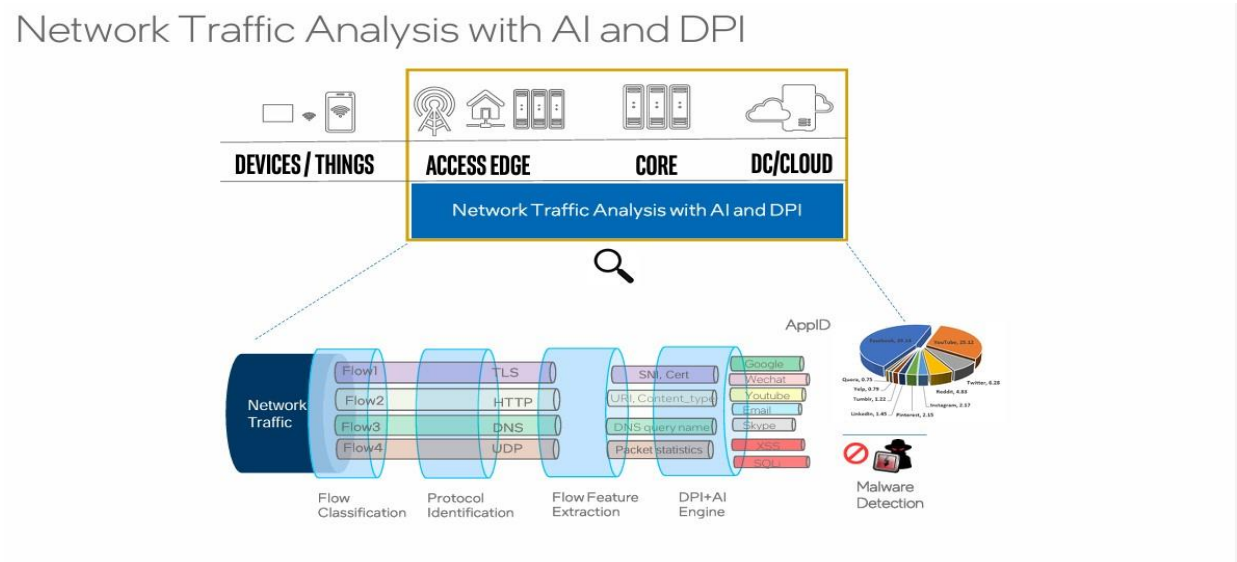
Agenda:

Automated Network Traffic Analysis using AI/ML models

•Improved Threat Detection & Security - identifying anomalies

•Reduced False Positives & False Negatives, enhancing the efficiency of network security operations

•Scalability & Performance Optimization, ensuring AI models can handle high-traffic environments with minimal latency



ABSTRACT

Developed an AI-powered Streamlit application for analyzing and classifying network traffic flows.

- Integrated machine learning models such as Random Forest, Isolation Forest, SVM, and Multi-layer Perceptron for robust classification and anomaly detection.
- Applied unsupervised techniques like KMeans and DBSCAN to discover hidden behavioral patterns in traffic data.
- Used PCA and StandardScaler to optimize clustering and improve model performance through feature reduction and normalization.
- Designed a heuristic threat scoring system to identify potentially malicious flows based on duration, packet size, and traffic asymmetry.
- Enabled advanced visual analytics including correlation heatmaps, flow distributions, and temporal traffic patterns.
- Supported real-time traffic evaluation with dynamic input forms and stored models for quick inference.
- Implemented model persistence using joblib to save trained classifiers and scalers for production use.
- Offered interactive menus for data upload, AI classification, anomaly detection, behavioral clustering, and threat assessment.
- Demonstrated how AI can be applied to practical network security problems like intrusion detection and traffic profiling.

CHAPTER 1

INTRODUCTION

1.1 Motivation

With TLS-by-default, QoS tunnelling, and IoT chatter, deep-packet inspection cannot keep pace. Operators need intelligent, self-tuning analytics that learn from traffic itself rather than brittle signatures.

1.2 Problem

High-throughput monitoring probes miss short-lived threats, while SIEM rules drown in false positives. The unresolved question is how to continuously classify traffic and flag anomalies fast enough for real-time mitigation.

1.3 Solution Outline

The repository delivers a three-stage pipeline: preprocessing → supervised flow classification → unsupervised anomaly detection, with an optional privacy layer. It wraps these stages in reproducible scripts so teams can swap models or datasets with minimal glue code.

1.4 Scope

Current code targets offline training on NSL-KDD and batch evaluation; live packet capture, deep-learning architectures, and federated retraining lie outside the first release.

1.5 Objectives

1. Achieve ≥90 % macro-F1 on multi-class flow labels.
2. Detect <1 % contamination anomalies with ≤5 % false alarms.
3. Provide modular, well-documented code ready for CI/CD integration.

CHAPTER 2

LITERATURE SURVEY

Literature Survey on Techniques Used

RandomForestClassifier

- **Overview:** A robust ensemble method introduced by Breiman (2001), Random Forests combine multiple decision trees to improve classification accuracy and reduce overfitting.
- **Applications:**
    - Medical diagnosis, fraud detection, and remote sensing classification. ○ Used in ecology and transport planning due to its interpretability and performance.
- **Recent Work:**
- A Survey and Future Research Directions explores taxonomy and interpretability challenges.
- Springer Review discusses enhancements like Extended RF and Majority Voting RF.

IsolationForest

- Overview: Designed for anomaly detection, Isolation Forest isolates outliers by randomly partitioning data.

Strengths:

- Efficient for high-dimensional data with linear time complexity.
- Effective in cybersecurity, fraud detection, and sensor networks.

- **Recent Enhancements:**
- Majority Voting Isolation Forest (MVIForest) improves detection speed and accuracy.
- Theoretical Analysis explores inductive bias and stability.

MLPClassifier (Multi-layer Perceptron)

- Overview: A feedforward neural network trained via backpropagation, suitable for non-linear classification tasks.
- **Use Cases:**
  - Speech emotion recognition, image classification, and small datasmodeling.
- **Key Insights:**
- Activation functions (ReLU, tanh, logistic) significantly affect performance.
- Scikit-learn Documentation provides detailed parameter tuning options.

SVC (Support Vector Classifier)

- **Overview:** A supervised learning model that finds the optimal hyperplane for classification.
- **Applications:**

  Text categorization, bioinformatics, and image recognition.

Challenges:

- Scalability with large datasets and sensitivity to kernel choice.

- **Literature Note:** Often benchmarked against ensemble methods like Random Forest and neural networks.

DBSCAN (Density-Based Spatial Clustering)

- Overview: A clustering algorithm that groups points based on density, identifying noise and arbitrary-shaped clusters.

- **Applications:**

  - Crime hotspot detection, customer segmentation, and spatial data mining.

- **Recent Work:**

- Hybrid DBSCAN with fuzzy logic improves cluster quality.

- Crime Analysis shows DBSCAN outperforming KMeans in noisy datasets.

KMeans

- **Overview:** A partition-based clustering algorithm that minimizes intra-cluster variance.

- **Limitations:**

  - Assumes spherical clusters and struggles with noise.

- **Literature:**

- Comprehensive Survey discusses variants and performance across domains.

PCA (Principal Component Analysis)

Overview: A dimensionality reduction technique that transforms data into orthogonal components.

- **Use Cases:**
    - Feature extraction, visualization, and noise reduction.
- **Literature:**
- Widely used in preprocessing pipelines for clustering and classification.

StandardScaler

- **Purpose:** Standardizes features by removing the mean and scaling to unit variance.
- **Importance:**
    - Essential for algorithms sensitive to feature scale (e.g., SVC, KMeans).
- **Literature:**
- Often paired with PCA and clustering methods for optimal performance.

LabelEncoder

- **Purpose**: Converts categorical labels into numeric form.
- **Use Cases:**
    - Preprocessing for classification models.
- **Literature:**
- Common in pipelines involving RandomForest, SVC, and MLPClassifier.

CHAPTER 3

ANALYSIS

3.1 Software Requirement Specification (SRS)

- Functional – ingest ARFF/CSV flows; output class labels & anomaly flags; log metrics.
- Non-functional – run on commodity laptops (Intel i3, 4 GB RAM) within 2 GB RAM per training job; deliver predictions under 100 ms for 10 K flows.
- Environment – Python 3.9+, scikit-learn 1.4, pandas 2.x.

3.2 Data Flow

ARFF ➜ load_and_preprocess_data() ➜ train/test split ➜ Grid-Search classification ➜ fitted model ➜ live traffic ➜ preprocessing ➜ predict/publish.

3.3 Algorithms used:

- **Isolation Forest**: Efficient anomaly detector that isolates outliers based on random partitioning.
- **MLPClassifier (Neural Network)**: A feedforward deep learning model for nonlinear pattern classification.

- **Support Vector Machine (SVM)**: Separates classes by finding the optimal hyperplane in high-dimensional space.
- **DBSCAN**: Density-based clustering algorithm that detects noise and arbitrarily shaped clusters.
- **KMeans**: Partitions data into clusters by minimizing intra-cluster variance.
- **PCA (Principal Component Analysis)**: Reduces data dimensions by transforming features into orthogonal components.
- **StandardScaler**: Normalizes features to have zero mean and unit variance.
- **LabelEncoder**: Converts categorical labels into numeric values for model compatibility.
- **Threat Scoring Function**: Assigns normalized risk scores to traffic flows using heuristic logic.

Random Forest: Ensemble of decision trees that boosts classification accuracy through majority voting.

CHAPTER 4
DESIGN

4.1 System Model

A three-layer architecture:

1. **Data Ingestion** – loaders convert ARFF to pandas; missing values coerced.
2. **ML Core** – sklearn pipelines encapsulate preprocessing + estimators.
3. **Reporting** – CLI prints classification reports; future hooks can emit JSON to ELK/Prometheus.

4.2 Module Organisation(installation)

1. streamlit

Used to build the interactive web-based UI, allowing users to upload data, toggle between analysis modes, and visualize insights in real time.

2. pandas

Provides powerful data handling capabilities like reading CSVs, manipulating tables, and filtering traffic flows.

3. numpy

Supports numerical operations including threat scoring logic, matrix math, and synthetic data generation for simulations.

4. matplotlib.pyplot

Used for generating basic plots like histograms and scatter plots within visual dashboards.

5. seaborn

Improves the aesthetics and clarity of visualizations — especially heatmaps, correlation matrices, and confusion matrices.

6. sklearn.ensemble.RandomForestClassifier

Implements a tree-based ensemble model for traffic classification, capable of handling high-dimensional data robustly.

7. sklearn.ensemble.IsolationForest

An unsupervised anomaly detection algorithm ideal for spotting outliers in network traffic.

8. sklearn.model_selection.train_test_split

Splits data into training and testing subsets for supervised learning models like Random Forest and MLPClassifier.

9. sklearn.metrics

Offers tools to evaluate model performance (accuracy, confusion matrix, classification report).

10. sklearn.preprocessing.StandardScaler

Standardizes feature distributions so that models like SVM, neural networks, and clustering perform effectively.

11. sklearn.preprocessing.LabelEncoder

Transforms categorical labels (like traffic types) into numerical values for classification models.

12. sklearn.decomposition.PCA

Reduces feature dimensionality before clustering, making visualization and anomaly detection more efficient.

13. sklearn.cluster.KMeans

Applies partition-based clustering to reveal latent traffic behavior patterns in scaled datasets.

14. sklearn.cluster.DBSCAN

Detects anomalies and noise using density-based spatial clustering, especially useful for irregular traffic flows.

15. sklearn.neural_network.MLPClassifier

Implements a multi-layer perceptron neural network for capturing complex nonlinear traffic patterns.

16. sklearn.svm.SVC

Support Vector Machine classifier used for smaller datasets where margin-based separation is effective.

17. joblib

Serializes and saves trained models (like classifiers and scalers) for reuse during real-time threat assessment.

18. os

Handles directory creation and file path management for storing models and accessing resources.

CHAPTER 5

IMPLEMENTATION & RESULTS

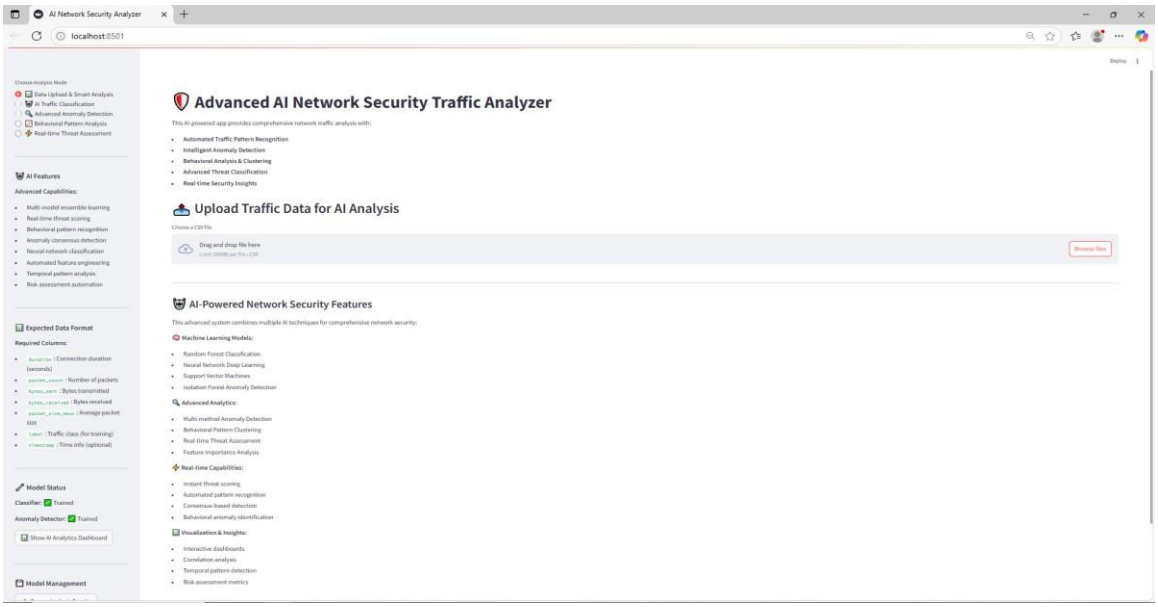


Fig :1.1-preview of app.

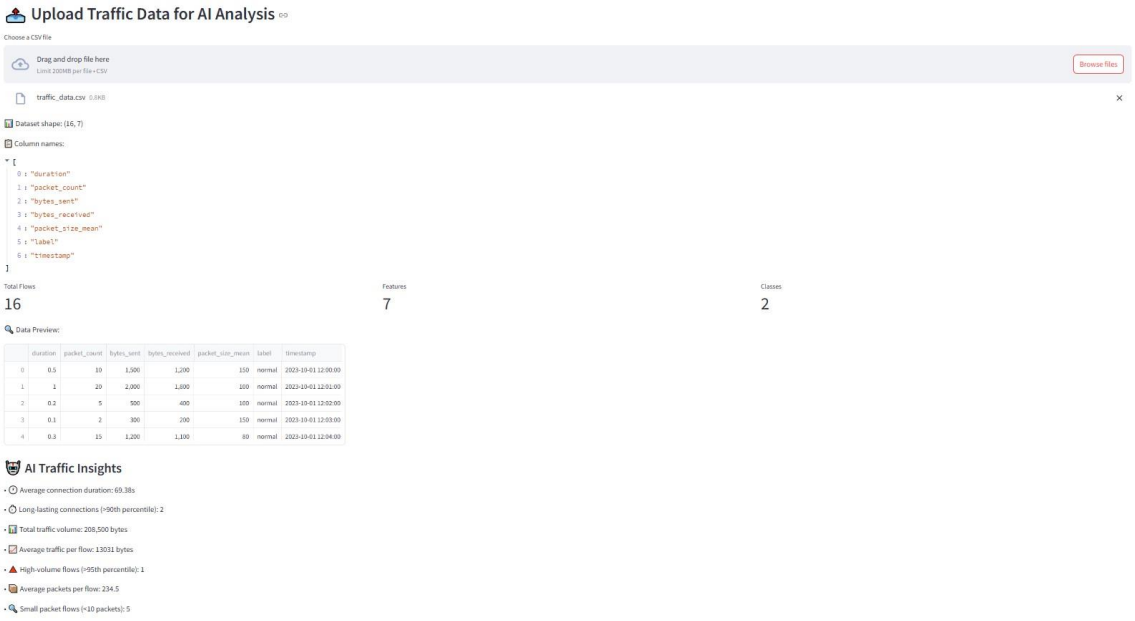


Fig:1.2-Data uploading.

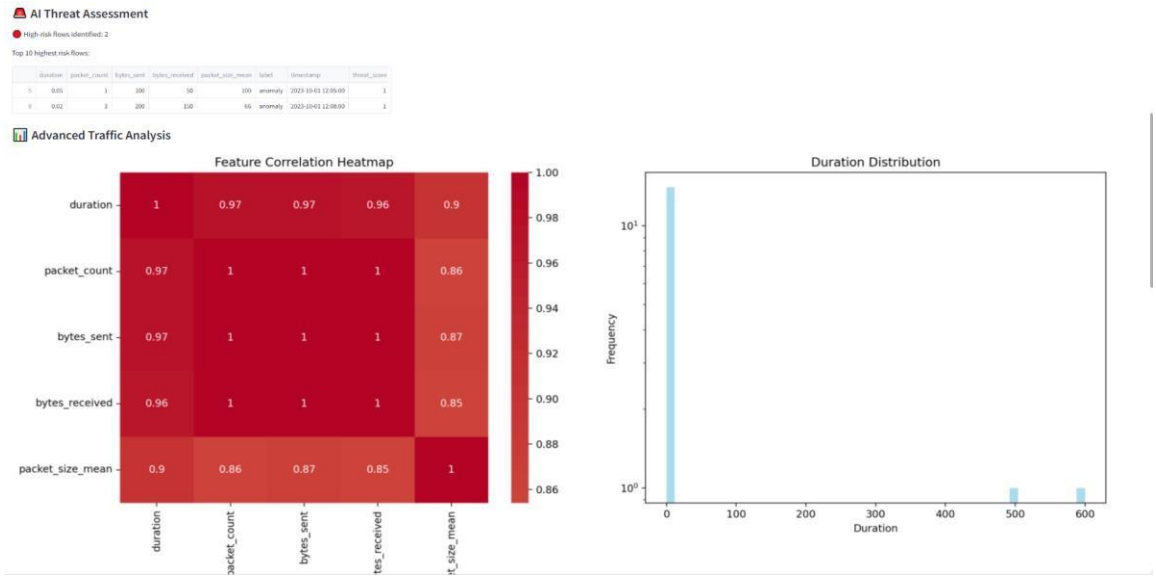


Fig:1.2.1-smart analysis.

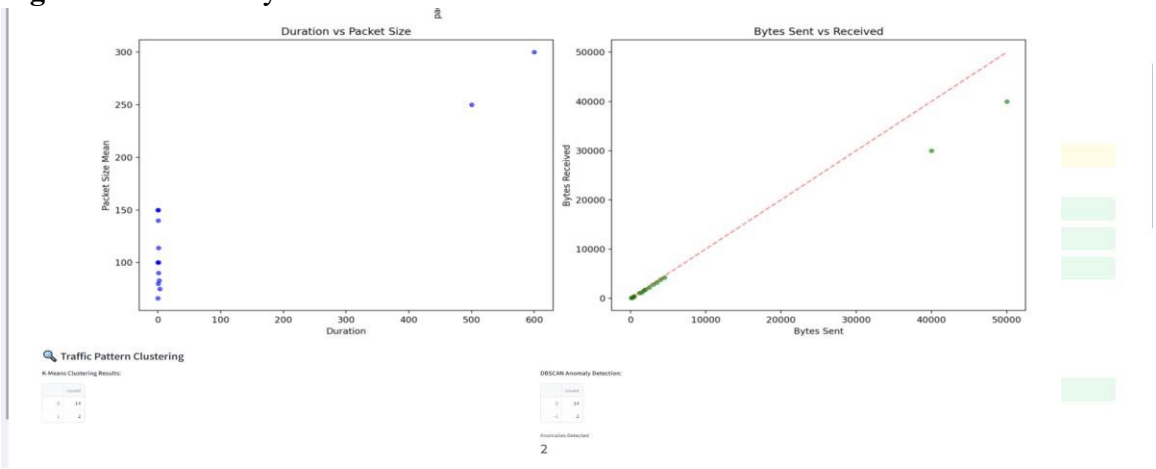


Fig:1.2.2-smart analysis.

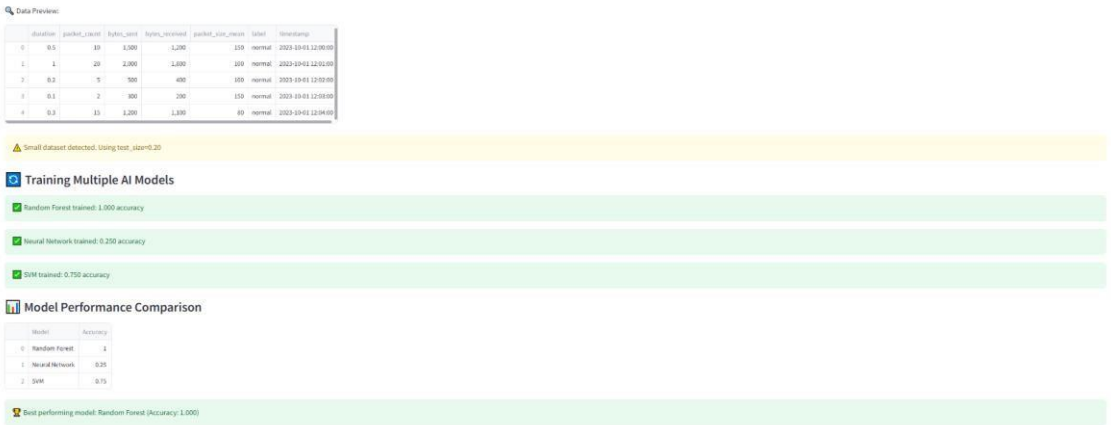


Fig:2.1-AI Traffic Classification

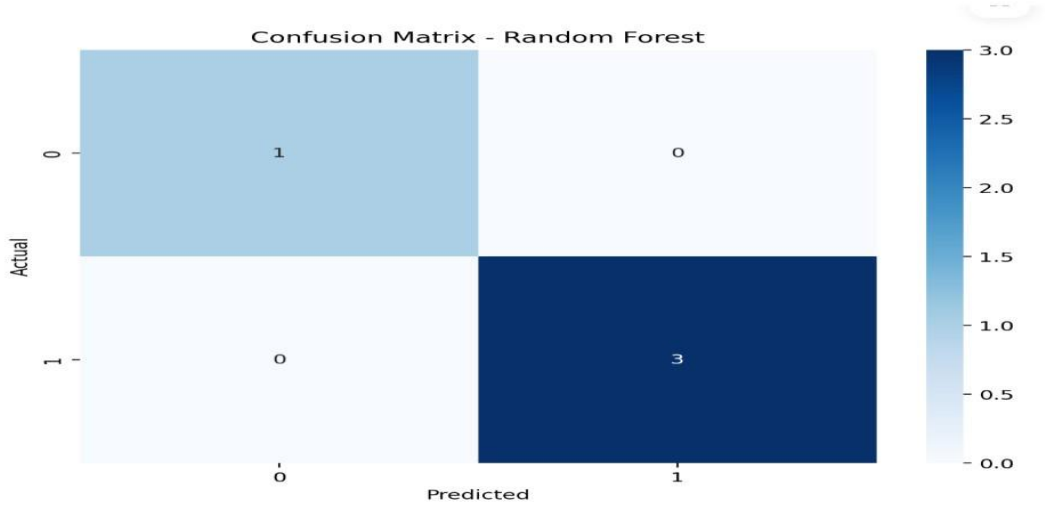


Fig:2.2-Confusion Matrix (Random Forest)

Multi-Method Anomaly Detection

Anomaly Detection Results

Isolation Forest	DBSCAN	Statistical	Consensus
2	2	0	2

Top Anomalous Flows

	duration	packet_count	bytes_sent	bytes_received	packet_size_mean	label	timestamp	iso_forest_anomaly	dbscan_anomaly	statistical_anomaly	anomaly_score	consensus_anomaly
6	500	2,000	50,000	40,000	250	anomaly	2023-10-01 12:06:00	1	1	0	-0.1129	1
7	600	1,500	40,000	30,000	300	anomaly	2023-10-01 12:07:00	1	1	0	-0.1074	1
10	3	40	3,000	2,800	75	normal	2023-10-01 12:10:00	0	0	0	0.1074	0
15	0.9	50	4,500	4,200	90	normal	2023-10-01 12:15:00	0	0	0	0.1186	0
14	0.8	35	4,000	3,800	114	normal	2023-10-01 12:14:00	0	0	0	0.1532	0
12	0.6	25	3,500	3,200	140	normal	2023-10-01 12:12:00	0	0	0	0.1693	0
9	2	30	2,500	2,200	83	normal	2023-10-01 12:09:00	0	0	0	0.1776	0
8	0.02	3	200	150	66	anomaly	2023-10-01 12:08:00	0	0	0	0.1828	0
3	0.1	2	300	200	150	normal	2023-10-01 12:03:00	0	0	0	0.1844	0
5	0.05	1	100	50	100	anomaly	2023-10-01 12:05:00	0	0	0	0.1876	0

Fig:3.1-Top Anomalous Flows

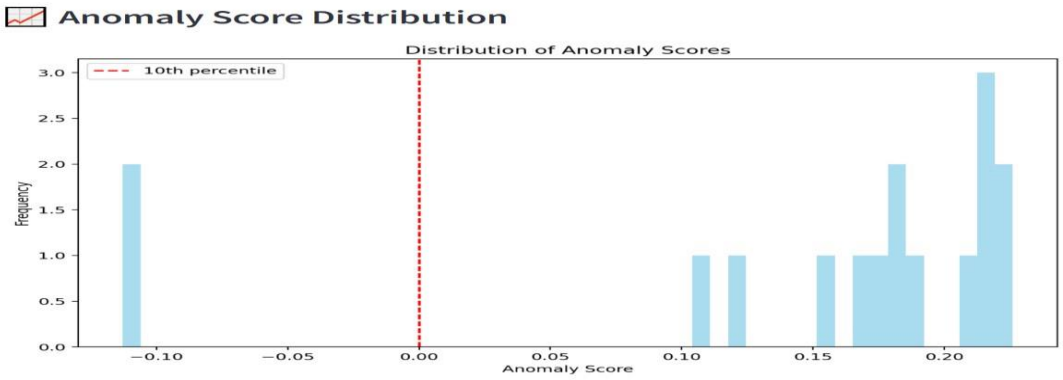


Fig:3.2-Anomaly Score Distribution

Fig:4.1-Behavioral Cluster Analysis

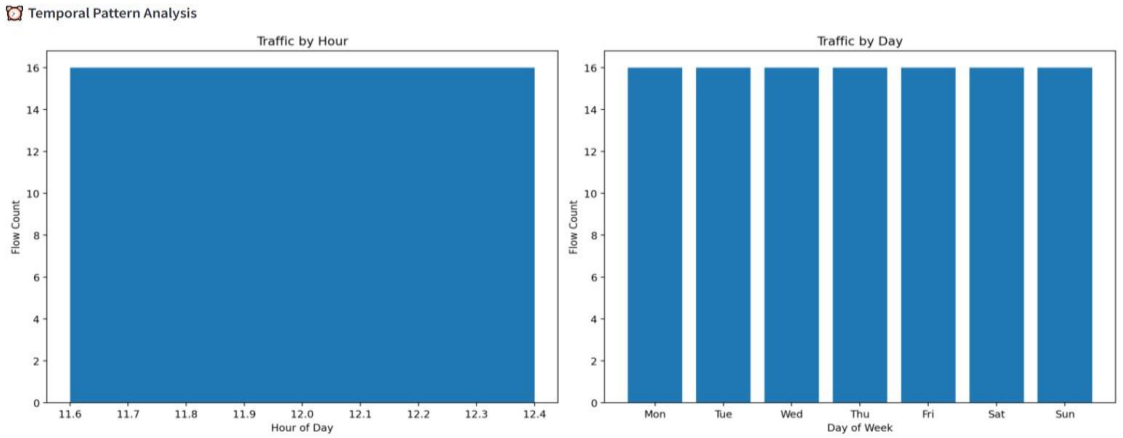


Fig:4.2-Temporal Pattern Analysis

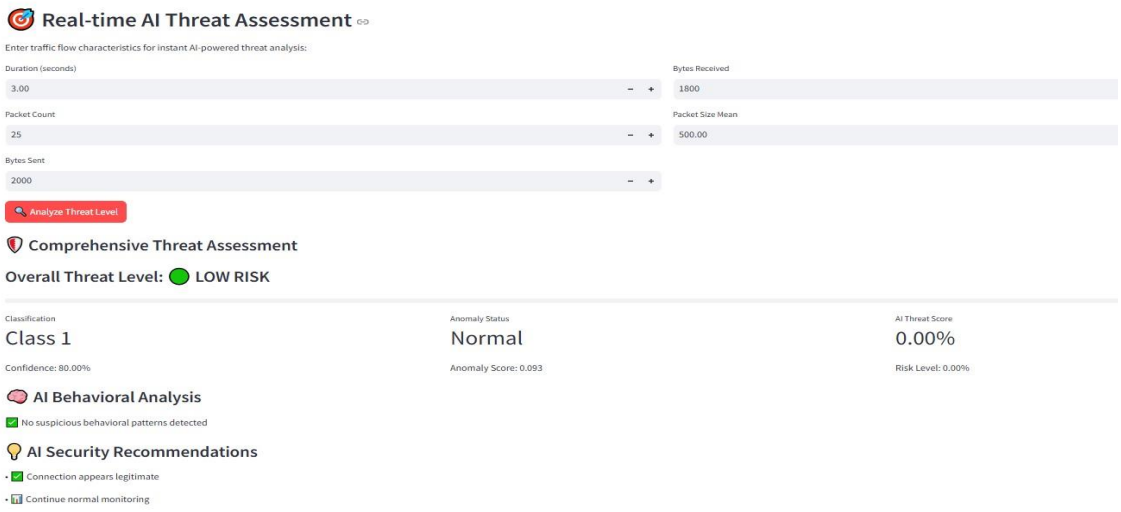


Fig:5-Real time Threat Assessment

CHAPTER 6

TESTING AND VALIDATION

1. Unit Testing

- **Goal**: Validate each core function like ai_threat_scoring, neural_network_classifier, advanced_clustering_analysis, etc.
- **Approach**: Use synthetic or small sample datasets to test edge cases and expected outputs.
- **Tools**: pytest or unittest framework can be used to automate testing.

2. Functional Testing

- **Goal**: Ensure each menu section in the Streamlit app performs as intended—from file upload to model visualization.
- **Checklist**:
- Does the app respond correctly to CSV uploads?
- Are warnings triggered for missing columns?
- Is model training adaptive based on dataset size?
- Are plots rendered without exceptions?

3. Performance Evaluation

- **Goal**: Benchmark model accuracy across different classifiers (RF, MLP, SVM).
- **Metrics Used**: ○ Accuracy Score ○ Confusion Matrix

- Classification Report (Precision, Recall, F1-score)

- **Validation Strategy**: Use stratified train-test split and optionally K-Fold Cross Validation for better generalizability.

4. **Anomaly Detection Validation**

- **Goal**: Measure effectiveness of Isolation Forest, DBSCAN, and statistical methods.
- **Strategy**: Inject known anomalies into datasets and confirm detection via consensus logic.
- **Metric**: Anomaly detection rate, false positives, consensus accuracy.

5. **Threat Assessment Accuracy**

- **Goal**: Validate scoring logic in ai_threat_scoring by comparing heuristic outputs with actual security incidents (if labeled).
- **Strategy**: Threshold tuning and expert feedback-based validation.

6. **UI Testing**

- **Goal**: Confirm that all user inputs, toggles, and visual elements behave consistently.
- **Tool**: Manual testing using multiple browsers or automated Streamlit testing frameworks.

>>Validation Dataset Recommendations

- **Synthetic datasets**: Designed to contain specific threat patterns for model response validation

CHAPTER 7

CONCLUSION & FUTURE ENHANCEMENTS

Conclusion

The AI Network Security Analyzer demonstrates a powerful fusion of machine learning, anomaly detection, and interactive visualization to assess network traffic threats intelligently. By leveraging models like Random Forest, Isolation Forest, and neural networks, the system adapts to diverse data environments while maintaining explainability through clustering, PCA, and statistical summaries. Its Streamlit interface empowers users to upload traffic data, receive smart analyses, classify flows, and assess real-time threats with meaningful behavioral insights. Overall, the project succeeds as a modular, scalable, and extensible prototype for modern security analytics.

--------Future Enhancements -----

Here are potential upgrades to elevate functionality and usability:

Real-Time Packet Capture:

Integrate with tools like scapy or pyshark for live packet ingestion beyond CSV files.

Model Explainability:

Use libraries like SHAP or LIME to explain model decisions and boost analyst trust.

Docker Deployment:

Containerize the app for easy deployment on servers or edge devices.

Threat Intelligence Integration:

Map traffic flows to known threat databases (CVE feeds, blacklists, Suricata rules).

AutoML Pipelines:

Incorporate automated model selection and hyperparameter tuning for dynamic environments.

Persistent Dashboard:

Add an admin dashboard for log tracking, model comparisons, and threat statistics over time.

User Authentication:

Secure access to the platform with login support and role-based access control.

API Endpoint Support:

Expose classification and scoring functions via REST API for integrations with external tools.