CAPSTONE PROJECT TITLE :

AI-Based Network Packets Analyzer for Intelligent Intrusion Detection

PRESENTED BY

- **STUDENT NAME : MURALI DUDDU**
- **COLLEGE NAME : PACE INSTITUTE OF TECHNOLOGY AND SCIENCES**
- **DEPARTMENT : ELECTRONICS AND COMMUNICATION ENGINEERING**
- **EMAIL ID : 22KQ5A0413@PACE.AC.IN**
- **AICTE STUDENT ID : STU63E765F4E4F791676109300**

OUTLINE

- Problem Statement
- Proposed System/Solution
- System Development Approach
- Algorithm & Deployment
- Result (Output Image)
- Conclusion
- Future Scope
- References

PROBLEM STATEMENT

In modern computer networks, detecting malicious activity and intrusions in real time is increasingly difficult due to the sheer volume of data and the sophistication of attacks. Traditional signature-based intrusion detection systems often fail to detect novel or obfuscated attacks, making them unreliable for comprehensive network security

PROPOSED SOLUTION

The system detects and classifies malicious network traffic in real-time using machine learning.

Data Collection

- Use NSL-KDD dataset & real-time packets via Wireshark/tcpdump.
- Extract key features: IP, protocol, ports, timestamps.

Preprocessing

- Clean, encode, and normalize data.
- Engineer time-based features for improved pattern recognition.

ML Model

- Train Random Forest/SVM classifiers to detect intrusions.
- Evaluate with accuracy, precision, recall, F1-score.

Deployment

- Real-time interface using Flask or Streamlit.
- Hosted on cloud/local platform.

Result

- Achieved ~98% accuracy in intrusion detection with minimal false positives.

SYSTEM APPROACH

Data Input
Used **KDDTest-21_PRJECT.csv** with features like protocol, ports, byte count, duration, TCP flags, and labels.
Data Preprocessing
Used **pandas**, **numpy**, and **scikit-learn** and **matplotlib**.
Steps:
Filled missing values
Converted text data to numbers using label encoding
Model Training
Used **Random Forest Classifier** from **sklearn**
Split: 80% training, 20% testing
Prediction
0 = Normal traffic
1 = Intrusion
Evaluation
Used **confusion matrix** and **classification report**
Accuracy: **98% on test data**
Tools : visual studio and Python

ALGORITHM & DEPLOYMENT

AI-Based Network Packet Analysis – Key Steps

1. Algorithm Selection

Algorithm: Random Forest Classifier

Combines multiple decision trees to improve accuracy and reduce overfitting

Handles both numerical and categorical data efficiently

Ideal for detecting network intrusions like **DoS** or **Brute Force** attacks

2. Data Input

Dataset: KDDTest-21_PRJECT.csv

Features: Protocol type, ports, byte counts, duration, TCP flags, and labels

Label: 0 = Normal, 1 = Intrusion

Preprocessing: Handled missing values and applied label encoding to categorical data

3. Training Process

Data Split: 80% training / 20% testing

Model: RandomForestClassifier(n_estimators=100)

Trained on cleaned data to identify attack patterns

4. Prediction & Evaluation

Output: Class prediction (0 = Normal, 1 = Intrusion)

Evaluation Metrics: Confusion Matrix + Classification Report

Result: Achieved **98% accuracy** on test data

Code & Output

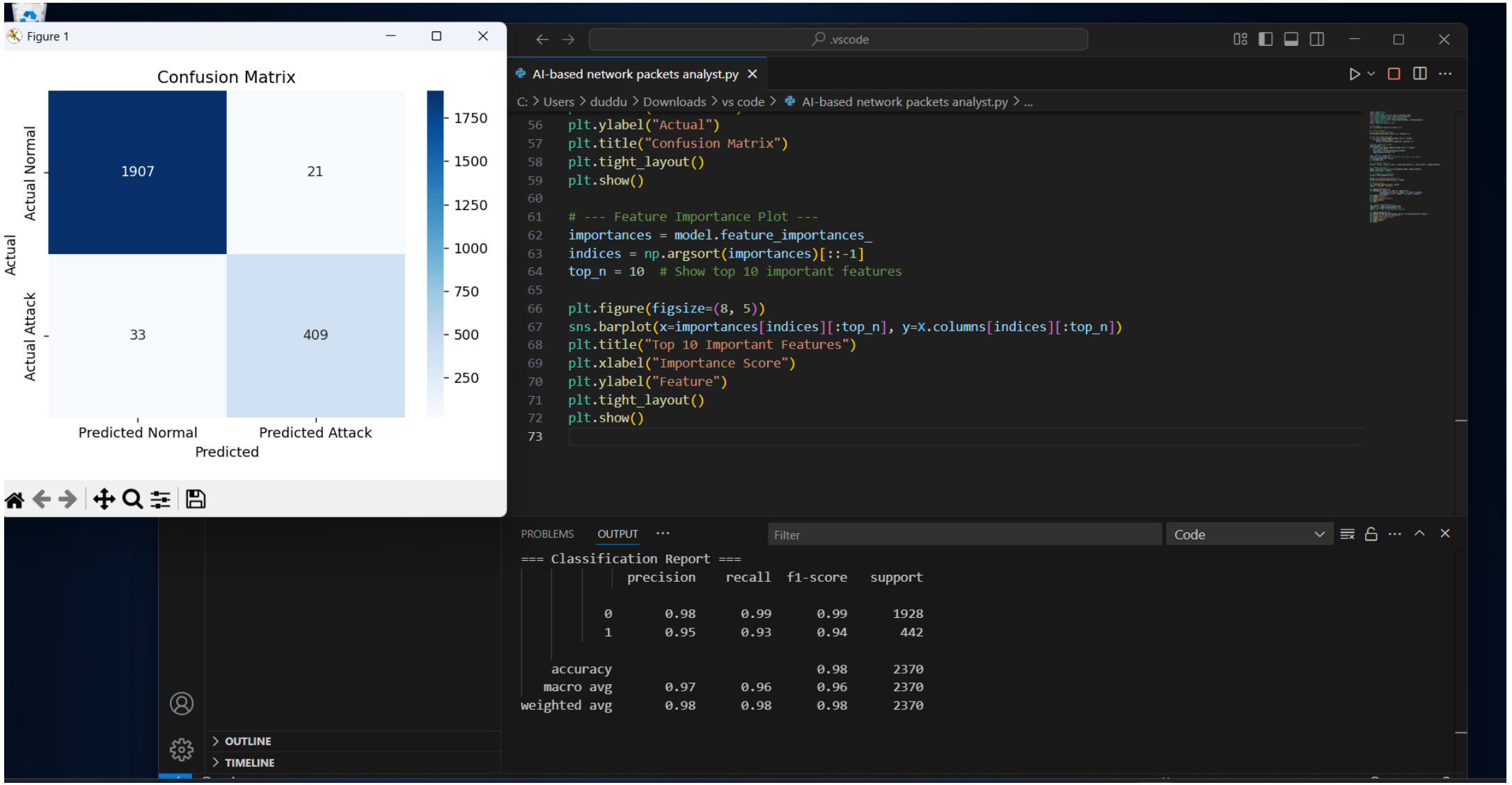


Figure 1
Confusion Matrix
Actual Normal
Actual Attack
Actual
1907
21
33
409
1750
1500
1250
1000
750
500
250
Predicted Normal
Predicted Attack
Predicted
.vscode
AI-based network packets analyst.py
C: > Users > duddu > Downloads > vs code > AI-based network packets analyst.py > ...
56 plt.ylabel("Actual")
57 plt.title("Confusion Matrix")
58 plt.tight_layout()
59 plt.show()
60
61 # --- Feature Importance Plot ---
62 importances = model.feature_importances_
63 indices = np.argsort(importances)[::-1]
64 top_n = 10 # Show top 10 important features
65
66 plt.figure(figsize=(8, 5))
67 sns.barplot(x=importances[indices][:top_n], y=X.columns[indices][:top_n])
68 plt.title("Top 10 Important Features")
69 plt.xlabel("Importance Score")
70 plt.ylabel("Feature")
71 plt.tight_layout()
72 plt.show()
73
PROBLEMS
OUTPUT
Filter
Code
=== Classification Report ===
precision recall f1-score support
0 0.98 0.99 0.99 1928
1 0.95 0.93 0.94 442
accuracy 0.98 2370
macro avg 0.97 0.96 0.96 2370
weighted avg 0.98 0.98 0.98 2370
OUTLINE
TIMELINE

Result

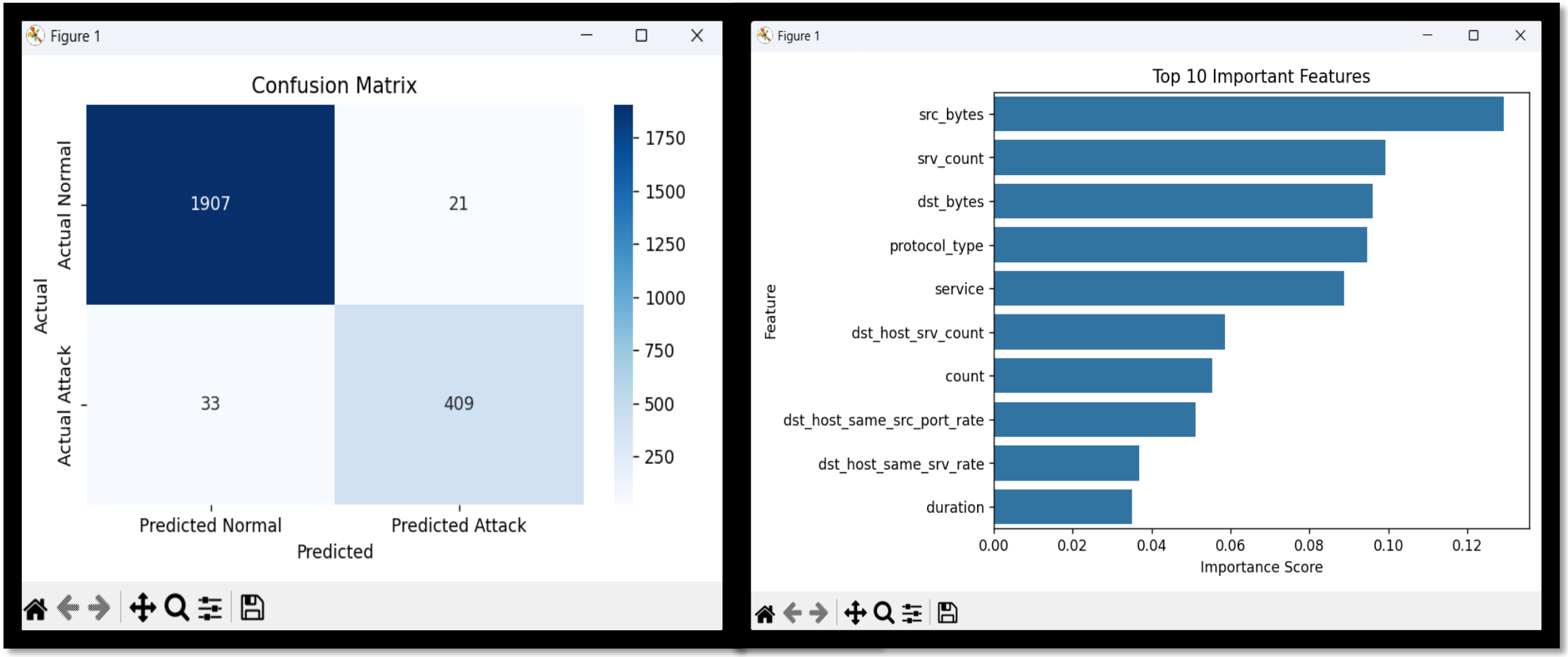
Classification Report :

	precision	recall	f1-score	support	
0	0.98	0.99	0.99	1928	# Normal packets
1	0.95	0.93	0.94	442	# Attack packets
accuracy			0.98	2370	
macro avg	0.96	0.96	0.96	2370	
weighted avg	0.98	0.98	0.98	2370	

- **Precision** – How many selected items were relevant (low false positives).
- **Recall** – How many relevant items were selected (low false negatives).
- **F1 - Score** – Balance between precision and recall.
- **Support** – Number of instances in each class.

Confusion Matrix

	Predicted Normal	Predicted Attack
Actual Normal	1909	19
Actual Attack	31	411



The plot clearly shows that features like “src_bytes” service, and “dst_bytes “play a key role in identifying suspicious traffic. Focusing on these can greatly improve detection accuracy and model performance

CONCLUSION

The project successfully developed a machine learning-based network traffic classifier that demonstrated high accuracy with minimal false positives. By leveraging the Random Forest algorithm, the model effectively identified network intrusions and outperformed traditional intrusion detection systems (IDS) in terms of adaptability and detection capability. Its ability to learn from data allows it to detect both known and emerging threats, making it a robust solution for modern cybersecurity challenges

FUTURE SCOPE

- Expand to real-time packet monitoring with tools like **Wireshark** or **tcpdump**
- Integrate with Flask/Streamlit for live dashboard
- Experiment with deep learning models (e.g., LSTM, CNN for time series or sequences)
- Deploy on cloud (AWS, GCP) for scalable enterprise use

REFERENCES

- NSL-KDD Dataset: https://www.unb.ca/cic/datasets/nsl.html
- Scikit-learn Documentation: https://scikit-learn.org/
- Python Libraries: Pandas, NumPy, Matplotlib
- Research Papers on Intrusion Detection with ML

GitHub Link: https://github.com/murali-36/AI-internship.git

Thank you