Intel Unnati Industry Project Report

# AI/ML for Network Security

By: Priyansh Pankaj (RA2311033010064)  
 Khushi Raghav (RA2311033010015)  
Date: 10th July 2025

# Abstract

In our interconnected digital world, cybersecurity has become a pressing concern. With more data being transmitted than ever before—often through encrypted channels—traditional security systems are struggling to keep up. The issue? These systems typically can’t see through encryption, making it hard to detect hidden threats.  
To tackle this, we initiated a project in Intel's Unnati program. The goal was to design an intelligent solution with Artificial Intelligence (AI) and Machine Learning (ML). Through training a model to differentiate between encrypted and unencrypted traffic based on network behavior, we envisioned providing an intelligent layer of protection. With Python in a Linux environment, we had trained a Random Forest model that was well over 90% accurate. This demonstrates how with appropriate tools, machine learning can be an effective weapon towards enhancing network security.

# Introduction

The internet is now ingrained in our day-to-day existence. From exchanging emails and participating in online meetings to online transactions and video streaming, we are dependent on digital connectivity every step of the way. But with dependence comes exposure.  
Classic network security software, which relies on preprogrammed rules, aren't able to deal with today's sophisticated and frequently encrypted threats. That is where machine learning comes in. Rather than following predetermined rules, it learns from actual data patterns. We were excited by the possibilities and wanted to create a model that could distinguish between encrypted and unencrypted traffic—by simply examining its structure and activity.

# 3. Methodology

3.1 Dataset Collection and Setup

We used a publicly available dataset that included both types of network traffic—encrypted and unencrypted. Features like entropy (randomness in data), number of packets, and packet size were key indicators.

3.2 Environment Configuration

We worked in a Linux-based environment (Ubuntu via WSL) using Python. Essential tools included:  
- pandas: for data processing  
- scikit-learn: for building the ML model  
- matplotlib: for visualization  
- joblib: to save the trained model

3.3 Data Preparation

The raw dataset needed some work. We filled in missing values, encoded categories into numbers (so the model could understand them), normalized numerical values, and then split the data into training and testing sets.

3.4 Model Building

We picked Random Forest because it’s not only accurate but also easy to understand. It builds many decision trees and combines their results.

3.5 Feature Interpretation

We wanted to know which data features had the most influence. A visual plot showed that entropy and packet statistics were major factors.

3.6 Performance Evaluation

To see how well the model worked, we used metrics like accuracy, precision, recall, and F1-score. This helped us understand not just how often it was right, but how reliable it was across different scenarios.

# 4. Optimization and Enhancements

Optimizing a model doesn't end once trained. We did the following to enhance performance:  
- Adjusted parameters such as number of trees and depth of trees  
- Dropped duplicate features that were redundant in information  
- Applied a random seed to ensure our results are reproducible  
- Corrected formatting errors that may lead to misinterpretation by the model

# 5. Results and Discussion

Our trained model gave us encouraging results:

- It achieved around 91% accuracy  
- Features like entropy, packet count, and size stood out as most important  
- We saved the model using joblib so it could be used in future projects or integrated into security systems

Overall, the project showed that machine learning can play a critical role in helping us detect threats hidden in encrypted traffic.

# 6. Code Snapshot

Below is the core Python script used for training, evaluating, and saving the model:

import pandas as pd  
import matplotlib.pyplot as plt  
from sklearn.model\_selection import train\_test\_split  
from sklearn.ensemble import RandomForestClassifier  
from sklearn.metrics import classification\_report, accuracy\_score  
import joblib  
  
df = pd.read\_csv("traffic.csv")  
label\_column = 'Label (0 = Unencrypted, 1 = Encrypted)'  
X = pd.get\_dummies(df.drop(label\_column, axis=1))  
y = df[label\_column]  
  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
model = RandomForestClassifier(n\_estimators=100, random\_state=42)  
model.fit(X\_train, y\_train)  
  
importances = model.feature\_importances\_  
sorted\_indices = importances.argsort()[::-1]  
  
plt.figure(figsize=(10, 6))  
plt.title("Feature Importance")  
plt.bar(range(len(importances)), importances[sorted\_indices], align="center")  
plt.xticks(range(len(importances)), X.columns[sorted\_indices], rotation=45, ha='right')  
plt.tight\_layout()  
plt.savefig("feature\_importance.png")  
plt.show()  
  
y\_pred = model.predict(X\_test)  
print("Accuracy:", accuracy\_score(y\_test, y\_pred))  
print("Classification Report:", classification\_report(y\_test, y\_pred))  
  
joblib.dump(model, "traffic\_model.joblib").

# 7. Team Collaboration

This was truly a team effort.

Khushi took the lead in handling and preparing the data, training the model, and summarizing its results. She also managed the report's structure and flow.

Priyansh focused on setting up the technical environment and visual tools. He also played a major role in tweaking the model for better performance.

Together, we worked in sync, complementing each other’s strengths, and ensuring everything came together smoothly.

# 8. Future Directions

There's a great deal more we can do with this base:

- Integrate the model with tools such as Scapy or Tshark to monitor live traffic   
- Create a web-based interface using Flask or Django for simple access  
- Investigate the use of deep learning models (such as LSTM or CNN) to analyze more sophisticated traffic patterns  
- Expand the model to identify particular forms of malicious activity  
- Attempt unsupervised learning to capture unknown or changing threats

# 9. Resources

- GitHub Repository:

<https://github.com/priyansh3105/AI-ML-for-networking-Intel-Unnati-Program>

- Project Demo:

<https://drive.google.com/file/d/19hEgjIKE99Ni3ZMtTYN69sLHpErMaygi/view>