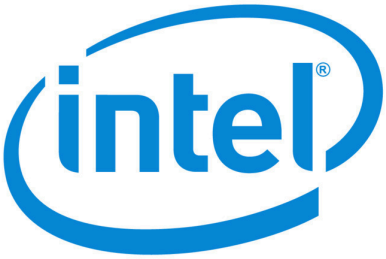
Intel® Unnati
Industrial Training Program 2025



Project Report
on
AI/ML For Networking

Submitted by
Network Traffickers
Natasha Gonsalves
Sanjeev Dronamraju

Manipal Institute of Technology, Manipal
Manipal, Karnataka 576104, India

Problem Statement: AI/ML for Networking

Introduction

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, applications, and enhance security in real-time, ensuring adaptive and intelligent network defense.

Objective

To build a supervised machine learning pipeline that can classify network traffic data as either benign or an attack, using the NSL-KDD dataset. The model is intended to be integrated into a Streamlit web app that supports both single-input and bulk prediction modes.

Tools & Technologies

- Programming Language: Python 3.12
- Libraries: pandas, numpy, scikit-learn, pickle, Streamlit

Deliverables

Threat Detection & Anomaly Identification Framework – AI-driven security mechanism to detect suspicious or malicious activity.

Important links:
GitHub - https://github.com/sanjeevd7/Intel_Unnati_Project
App Deployed - https://networkanalyse.streamlit.app/

Attacks Covered

The NSL-KDD dataset contains various network intrusions grouped into four categories:

- Denial of Service (DoS): Attacks aimed at overwhelming or crashing a system.

  DoS attacks included in this dataset are back, land, neptune, pod, smurf, teardrop, apache2, mailbomb, processtable, udpstorm.

- Probe: Attempts to gather information about a system or network.

  Probe attacks included are ipsweep, nmap, portsweep, satan, mscan, saint.

- User to Root (U2R): Exploits where a normal user tries to gain root privileges.

  U2R attacks included: buffer_overflow, loadmodule, perl, rootkit, xterm, ps.

- Remote to Local (R2L): Tries to gain local access from a remote machine.

  R2L attacks included: ftp_write, guess_passwd, imap, multihop, phf, spy, warezclient, warezmaster, named, sendmail, snmpguess, snmpgetattack, xlock, xsnoop, worm.

These categories help in understanding the intent and mechanism of each attack

Dataset Description

DATASET: NSL - KDD (Source)

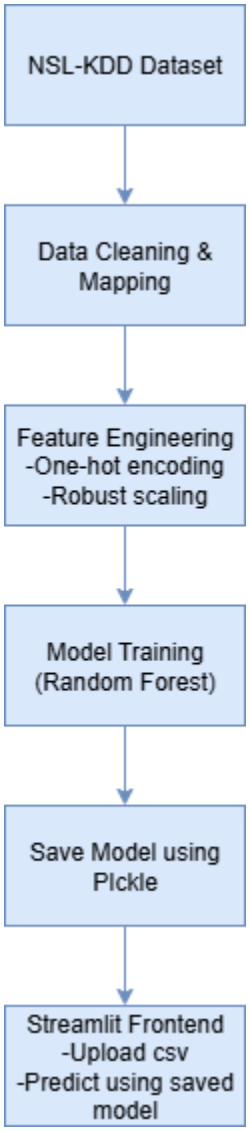
The dataset chosen for this project is the NSL-KDD dataset, which is a refined version of the original KDD Cup 1999 dataset. It addresses some of the key issues found in the original dataset, such as redundancy and class imbalance, making it more suitable for evaluating intrusion detection systems using machine learning.

The dataset includes various simulated network connection records, each labeled as either normal or one of several types of attacks. Each record consists of 41 features that describe attributes such as protocol type, service, flag, duration, number of failed logins, and other statistical measures extracted from raw network traffic.

Each connection record includes:

- Categorical features (e.g., protocol_type, service, flag) that describe the type of connection.
- Numerical features (e.g., duration, src_bytes, dst_bytes) that quantify behavior over the connection.
- Class labels that indicate whether the record is normal or belongs to one of the four main attack categories: DoS, Probe, R2L, U2R.

Architecture



Coding Structure

I. Data Preprocessing:

The dataset being used, i.e. the NSL-KDD dataset is loaded using Pandas. Column names are assigned and binary labels are mapped based on the attacks (benign v/s malicious).

One-hot encoding is applied to categorical features like protocol_type, service, and flag as they possess textual data.

RobustScaler has been used to scale the numeric features as it is less sensitive to outliers than standard scaling. This helps in normalizing feature ranges for better model performance.

II. Model Training:

Random Forest (RF) has been the preferred model selection due its high accuracy and low latency compared to other models.

Once the data is preprocessed, the dataset is separated into:
Features (X): Inputs to the model (all network attributes)
Target (y): What the model should learn to predict (attack_state)

Then the model is trained using the *RandomForestClassifier()*.

III. Model Testing and Evaluation:

After the model has been trained, it is tested using the test dataset. Metrics like Accuracy, Recall Score, Precision and F1 Score are evaluated. Confusion matrix is also plotted for better evaluation of the model.

GridSearch() has been to get the best RandomForest model and retain it. The best RF model, along with the scaler and the column list are saved as Pickle files (.pkl)

IV. Streamlit Integration:

The Streamlit application allows the user to visualise the project. It supports CSV file upload, therefore, the user can input CSV files with network packets that need to be inspected.

The CSV file is preprocessed, its columns are reordered and data is scaled using the same Scaler to match the training order.

The application displays the prediction results for each network packet query predicting if it's Normal or Malicious

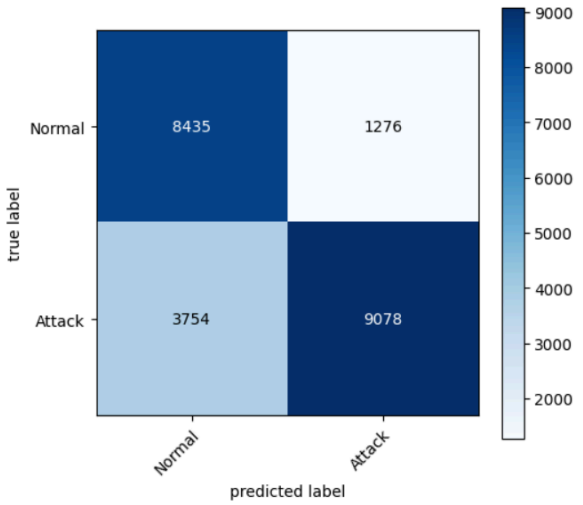
Results and Discussion

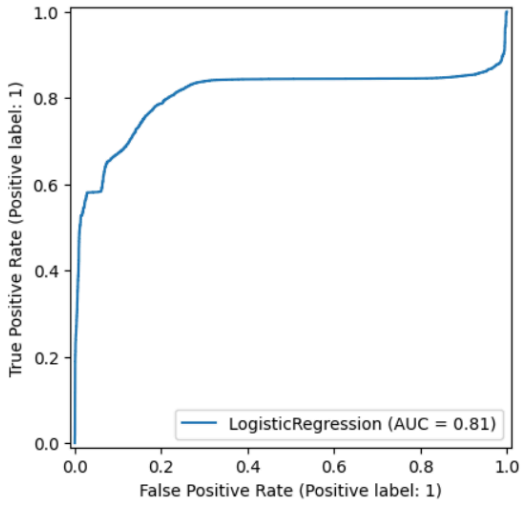
Different machine learning algorithms were tested to evaluate their performance in classifying network traffic as benign or malicious. Each model was assessed using the same preprocessed NSL-KDD dataset to ensure a fair comparison. The evaluation metrics considered i.e. the **Confusion Matrix** and **ROC Curve** of each algorithm was as follows:

1. **Logistic Regression**

   Logistic Regression is a simple and interpretable linear model. However, due to the complexity and non-linearity of network traffic patterns, its performance was moderate.

```
-------------------------------------------------

The Logistic Regression Model Accuracy   = 0.777

The Logistic Regression Model Sensitvity = 0.707

The Logistic Regression Model Precision  = 0.877

The Logistic Regression Model F1 Score   = 0.783

The Logistic Regression Model Recall     = 0.707

-------------------------------------------------
```





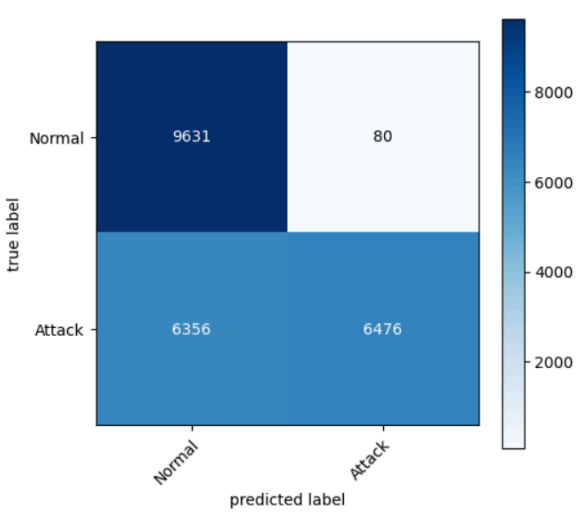
Confusion Matrix ROC Curve

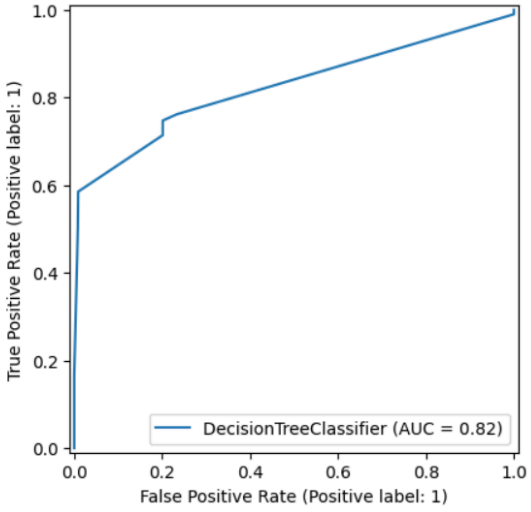
- Logistic Regression is linear; network data involves **complex, non-linear patterns**, which it can't capture well.
- Overlaps in feature space (e.g., similar packet lengths in benign and malicious traffic) make linearly separating them ineffective.

2. **Decision Tree Classifier**

   Decision Trees can capture non-linear relationships and feature interactions. While being very precise, this model actually behaved worse than linear regression.

```
-------------------------------------------------

The Decision Tree Classifier Model Accuracy   = 0.715

The Decision Tree Classifier Model Sensitvity = 0.505

The Decision Tree Classifier Model Precision  = 0.988

The Decision Tree Classifier Model F1 Score   = 0.668

The Decision Tree Classifier Model Recall     = 0.505

-------------------------------------------------
```





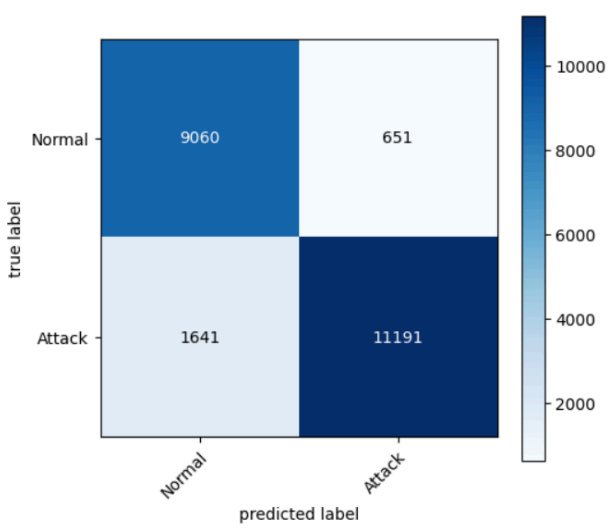
Confusion Matrix ROC Curve

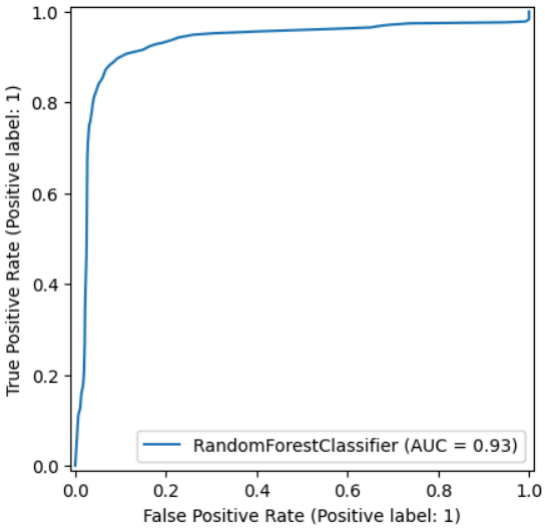
- Decision Trees tend to **overfit**, especially with many features (NSL-KDD has 41).
- It learns noise rather than generalizable patterns.
- Performs worse than logistic regression because of instability and lack of regularization.

3. Random Forest Classifier

Random Forest, an ensemble of Decision Trees, delivered the best performance due to its ability to reduce overfitting and handle a large number of features.

```
-------------------------------------------------

The Random Forest Classifier Model Accuracy   = 0.898

The Random Forest Classifier Model Sensitvity = 0.872

The Random Forest Classifier Model Precision  = 0.945

The Random Forest Classifier Model F1 Score   = 0.907

The Random Forest Classifier Model Recall     = 0.872

-------------------------------------------------
```



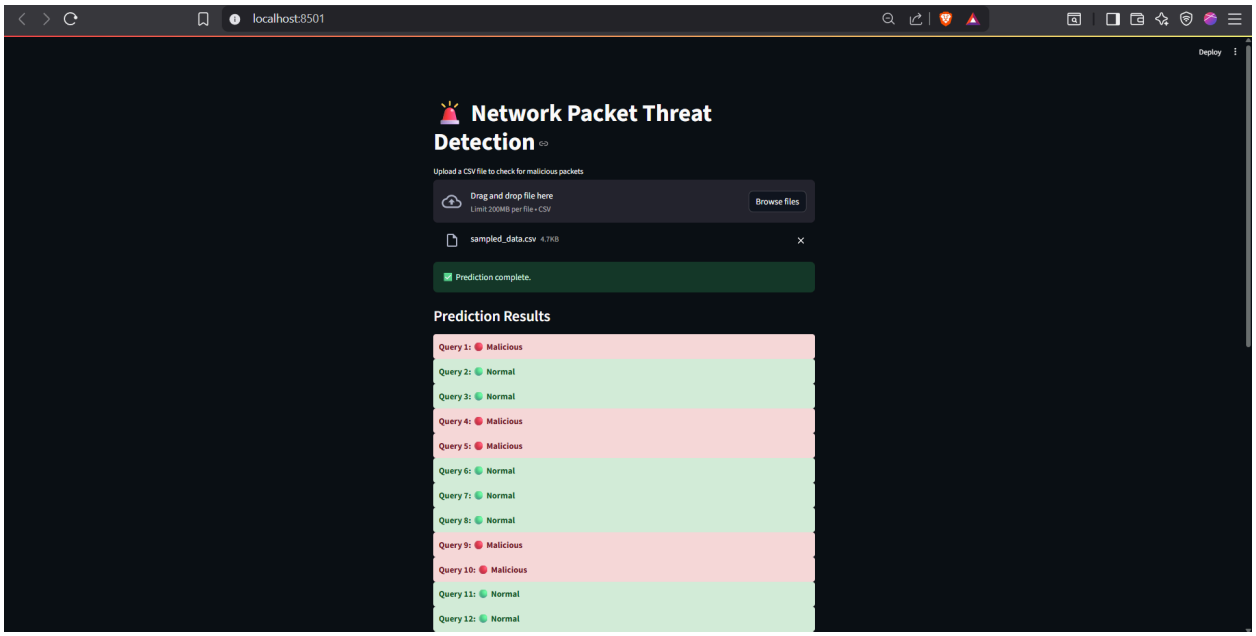


Confusion Matrix ROC Curve

- Combines multiple trees to reduce overfitting.
- Handles **non-linearity**, **feature interaction**, and **noise** robustly.
- Performs well even with categorical + numerical data (after encoding and scaling).
- However, it's still a **binary classifier**, so it may miss rare attack patterns.

The **Random Forest classifier** outperformed all other models in both accuracy and consistency. It handled feature interactions and non-linearities effectively, which are crucial for detecting complex and subtle patterns in network traffic.

For the Streamlit application, once the sample data was fed as a .csv file, the trained model predicted the nature of each query and shared the results as follows:



When cross referenced with the actual label of each query it was found that the model was able to predict most attacks accurately. However, it struggled to correctly predict a few specific attack types, particularly:

- snmpgetattack
- snmpguess
- processtable
- httptunnel

These attacks were often misclassified as Normal, leading to false negatives, which are critical in the context of security applications.

Possible Reasons for Misclassification:

1. Class Imbalance in the Dataset:
   These attack types have very few samples in the NSL-KDD dataset compared to common attacks like smurf, neptune, or portsweep. Random Forest models are data-hungry, and insufficient training samples for rare

attacks lead to poor generalization and low detection sensitivity.

2. Feature Overlap with Benign Traffic:
   Attacks like httptunnel and processtable may closely resemble normal traffic patterns in terms of the extracted feature values. If their statistical or protocol-level characteristics don't differ significantly from normal connections, the model struggles to distinguish them.

3. Binary Classification Limitation:
   Since the model is trained only to predict Normal vs Malicious, it doesn't explicitly learn the nuances between individual attack types. This can cause it to ignore subtle anomalies that deviate only slightly from benign patterns.

The above issues could be mitigated by using a multilabel-trained model, with a class-balanced approach like SMOTE.

Relevant Use Cases

- Enterprise Network Security: Real-time monitoring and detection of suspicious activities in corporate networks.

- Cloud Service Monitoring: Identifying unusual traffic patterns in cloud infrastructure to prevent service disruption.

- IoT Device Protection: Enhancing the security of IoT devices by classifying anomalous behavior.

- Intrusion Detection Systems (IDS): As a component in larger IDS solutions, the model can assist in early-stage filtering and classification.

- Academic Research: The pipeline can serve as a base model for further experimentation with deep learning or federated learning approaches.

Further Work

- Use DL for better prediction.
- Add performance visualization.
- Extend to multi-class classification.

Conclusion

The project demonstrates the effectiveness of AI/ML models, particularly Random Forest, in detecting and classifying network traffic as benign or malicious. With relevant preprocessing, hyperparameter tuning, and a user-friendly Streamlit interface, the solution provides a practical approach to threat detection. The promising results highlight the potential for deploying AI-driven security frameworks in real-world scenarios, paving the way for further innovations in intelligent network defense.

Contributions

To ensure equal contribution, the project responsibilities were evenly divided:

Sanjeev

- Data preprocessing and exploratory analysis.
- Implemented one-hot encoding and scaling manually.
- Built and trained the Random Forest Classifier.
- Created the model pipeline and tested it with various inputs.

Natasha

- Designed and implemented the Streamlit web application.
- Integrated the model with input handling for CSV features.
- Handled model saving/loading using pickle.
- Handled the documentation and testing of the final interface.

Both team members collaborated equally in debugging, tuning, evaluation, and writing the report.

Acknowledgements

We would like to express our sincere gratitude to the Intel® Unnati Industrial Training Program for providing us with this opportunity.

We also thank our mentors and coordinators for their continuous support and guidance throughout this project.

Finally, we acknowledge the creators of the NSL-KDD dataset and the open-source community whose tools made this work possible.