Network Intrusion Detection using Machine Learning

Project Report

Author: Shivangi Singh
Date: 8th April 2025

1. Executive Summary

This report summarizes a project focused on developing and evaluating machine learning models for network intrusion detection. Using the Kaggle "Network Intrusion Detection" dataset, the primary objective was to classify network connections as either 'normal' or 'anomaly'. Data pre-processing involved handling categorical features, scaling numerical features, and splitting the data. Several classification algorithms, including Decision Tree, Random Forest, and Gaussian Naive Bayes, were trained and evaluated. The Random Forest classifier demonstrated the highest performance, achieving over 99% accuracy, precision, recall, and F1-score on the held-out test set, indicating its strong potential for identifying network threats within this dataset.

2. Introduction

2.1 Problem Statement

Network security is paramount in today's interconnected world. Malicious activities, ranging from denial-of-service attacks to unauthorized access attempts, pose significant threats to data integrity, confidentiality, and system availability. Network Intrusion Detection Systems (NIDS) are crucial tools for identifying such threats in real-time or through traffic analysis. Traditional signature-based NIDS can struggle against novel or polymorphic attacks, creating a need for adaptive, learning-based approaches.

2.2 Objective

The primary objective of this project was to apply machine learning techniques to build robust classifiers capable of distinguishing between normal network traffic and various types of network intrusions (anomalies). This involved:

- Preprocessing network connection data containing diverse features.
- Training multiple machine learning models.
- Evaluating model performance using standard classification metrics.
- Identifying the most effective model and key predictive features within the context of the provided dataset.

3. Dataset Description

The project utilized the "Network Intrusion Detection" dataset sourced from Kaggle (https://www.kaggle.com/datasets/sampadab17/network-intrusion-detection). The analysis focused on the Train_data.csv file, which contains 25,192 records (network connections). Each record has 41 potential features describing the connection (e.g., duration, protocol, service, flags, byte counts, connection rates) and a binary 'class' label indicating 'normal' or 'anomaly'.

4. Methodology

The project followed a standard machine learning workflow implemented in a Jupyter Notebook:

4.1 Data Preprocessing

1. **Loading:** The dataset was loaded using the Pandas library.
2. **Target Encoding:** The binary target variable 'class' ('anomaly', 'normal') was encoded into numerical values (0, 1 respectively) using LabelEncoder.
3. **Feature Identification:** Features were automatically segregated into numerical (38 features) and categorical (3 features: protocol_type, service, flag).
4. **Scaling:** Numerical features were scaled using StandardScaler to have zero mean and unit variance, essential for distance-sensitive algorithms.
5. **Encoding:** Categorical features were converted into numerical format using OneHotEncoder. This expanded the feature space, resulting in approximately 122 features after preprocessing (exact number depends on unique categories encountered).
6. **Train/Test Split:** The dataset was split into a 70% training set (17,634 samples) and a 30% testing set (7,558 samples). Stratification based on the target variable was used to ensure both sets maintained the original class distribution (~46.6% anomaly, ~53.4% normal).

4.2 Model Selection and Training

Three different classification algorithms were selected to compare performance:

- **Decision Tree:** A simple, interpretable tree-based model.
- **Random Forest:** An ensemble method using multiple decision trees to improve robustness and reduce overfitting. Known for strong performance on tabular data. class_weight='balanced' was used to handle potential class imbalance.
- **Gaussian Naive Bayes:** A probabilistic classifier based on Bayes' theorem with an assumption of feature independence.

Models were trained on the preprocessed training data (X_train_processed, y_train).

4.3 Evaluation Metrics

Model performance was evaluated on the unseen test set (X_test_processed, y_test) using the following standard metrics, calculated using scikit-learn:

- **Accuracy:** Overall percentage of correct predictions.
- **Precision:** Ability of the classifier not to label as positive a sample that is negative (relevant for minimizing false alarms). Calculated using a weighted average.
- **Recall (Sensitivity):** Ability of the classifier to find all the positive samples (relevant for catching actual intrusions). Calculated using a weighted average.

- **F1-Score:** The harmonic mean of Precision and Recall, providing a single score balancing both. Calculated using a weighted average.
- **Confusion Matrix:** A table visualizing correct and incorrect predictions per class.
- **Classification Report:** Provides per-class precision, recall, and F1-score.

5. Results and Discussion

5.1 Model Performance Comparison

The performance of the trained models on the test set is summarized below:

Table 1: Model Performance Summary (Test Set)

Model	Accuracy	Precision (Weighted)	Recall (Weighted)	F1-Score (Weighted)
Decision Tree	0.9875	0.9875	0.9875	0.9875
Random Forest	**0.9921**	**0.9921**	**0.9921**	**0.9921**
Gaussian Naive Bayes	0.9653	0.9658	0.9653	0.9651

5.2 Key Findings

- **Best Performing Model:** The Random Forest classifier consistently outperformed the other models across all metrics, achieving an exceptional F1-Score of approximately 0.9921 on the test data.
- **High Accuracy:** All models performed relatively well, indicating that the features in the dataset are highly predictive of network intrusions. However, Random Forest demonstrated superior capability in correctly classifying both 'normal' and 'anomaly' instances.
- **Generalization:** The Random Forest model showed only a minor difference between training accuracy (~0.9998) and test accuracy (0.9921), suggesting good generalization and minimal overfitting.
- **Feature Importance (from Random Forest):** Analysis indicated that features related to connection flags (e.g., 'flag_SF'), service types (e.g., 'service_http', 'service_private'), connection rates (count, srv_count), and byte counts (src_bytes, dst_bytes) were among the most influential in distinguishing between normal and anomalous traffic. *(Optional: Insert small feature importance bar chart here if space allows)*.

5.3 Discussion

The results strongly suggest that machine learning, particularly ensemble methods like Random Forest, can be highly effective for network intrusion detection using connection-level features. The high performance achieved on this specific dataset highlights the quality of the features provided. However, performance in real-world scenarios might differ due to concept drift, evolving attack patterns, and variations in network configurations not captured in this static dataset.

6. Conclusion

This project successfully demonstrated the application of machine learning for network intrusion detection. By preprocessing the Kaggle NIDS dataset and training models like Decision Tree, Random Forest, and Gaussian Naive Bayes, we were able to achieve high classification performance. Random Forest emerged as the most effective model, yielding F1-scores exceeding 99% on the test set. The findings underscore the potential of data-driven approaches for enhancing network security.

7. Future Work

Potential directions for future work include:

- Evaluating models on more diverse and larger NIDS datasets (e.g., CIC-IDS2017/2018).
- Exploring deep learning models (e.g., LSTMs, CNNs) for potentially capturing temporal patterns or complex feature interactions.
- Investigating techniques for handling real-time data streams and model updates.
- Performing more granular multi-class classification to identify specific attack types instead of just binary anomaly detection.