Network Traffic Anomaly Detection Using Machine Learning: Identifying Cyber Threats from UNSW-NB15 Dataset

Students:
Ivan Leonychev T00727314
Polina Kniazeva T00704993

Course: ADSC 4710 - Machine Learning in Applied Data Science

Instructor: Minoli Munasinghe, MSc Data Science, BSc Applied Statistics

Submission Date: March 28, 2025

Project Abstract:

With the exponential increase in digital reliance and internet usage, cyber threats have escalated significantly, posing critical challenges to network security. Timely and accurate detection of network traffic anomalies is essential to mitigate cyber-attacks effectively.

This project addresses the challenge of anomaly detection by implementing advanced machine learning techniques on the UNSW-NB15 dataset, which consists of extensive records of network packets and flow data covering both normal traffic and various attack scenarios such as denial-of-service, reconnaissance, and exploits.

Comprehensive preprocessing included feature engineering (packets per second, bytes per second), handling missing and infinite values, and transforming data through scaling and encoding techniques.

Two models, Random Forest and XGBoost classifiers, were developed and rigorously evaluated using multiple metrics (accuracy, precision, recall, F1-score).

Results indicated that both models performed robustly, with the XGBoost classifier demonstrating superior performance (around 90% accuracy). Effective feature engineering notably enhanced model accuracy.

Challenges encountered included avoiding data leakage and computational limitations. The project's successful implementation lays a solid foundation for future enhancements, such as employing deep learning approaches and exploring real-time anomaly detection scenarios.

1. Introduction

With the exponential increase in internet usage and digital reliance, cyber-attacks pose a significant threat to the security of data and IT infrastructures globally. Network security, therefore, plays a crucial role in safeguarding sensitive information and ensuring operational continuity.

Detecting network traffic anomalies efficiently remains a significant challenge due to the complexity, volume, and evolving nature of cyber threats. Traditional methods often fail to identify subtle or novel threats, necessitating advanced machine learning approaches to enhance detection capabilities.

This project aims to build an accurate and efficient machine learning model to detect anomalies in network traffic, distinguishing between benign and malicious activities. Leveraging quantitative network flow features provided by the UNSW-NB15 Dataset, the model will classify network packets or flows as normal or anomalous (potentially malicious), thereby enhancing cybersecurity capabilities and contributing to proactive threat mitigation.

Objectives:

- To implement effective machine learning algorithms for predicting network anomalies.
- To compare the performance of selected machine learning models.

2. Dataset & Preprocessing

The UNSW-NB15 dataset was developed by the Australian Centre for Cyber Security (ACCS). It contains network packet and flow data, capturing various attack scenarios (e.g., exploits, DoS, generic, reconnaissance attacks) alongside normal network traffic. The dataset includes quantitative and categorical attributes relevant to cybersecurity applications.

Dataset Structure:

- Training Set: Used for building and training machine learning models.
- Testing Set: For evaluating the model's accuracy and generalization.

- Additional CSV files include detailed feature descriptions and attack category definitions.

Key Features:

- Quantitative Network Traffic Attributes such as:
  - Flow duration and byte counts
  - Source and destination port numbers
  - Number of packets and packet size statistics
  - Time-based statistical metrics (e.g., inter-arrival times)
- Protocol-specific features (TCP, UDP, ICMP, etc.)
- Categorical Information:
  - Attack labels: Indicates whether the record represents normal or anomalous (attack) traffic.
  - Attack categories: Types of attacks (e.g., DoS, Exploits, Generic, Reconnaissance).
- Attack Categories (examples):
  - Exploits: Attempts to exploit vulnerabilities.
  - DoS (Denial-of-Service): Overwhelming resources to make services unavailable.
  - Reconnaissance: Information-gathering attempts.
  - Generic Attacks: General malicious traffic not specifically categorized.

Dataset Size: Approximately 2.5 million records with around 49 features per record.

The dataset was sourced from **Kaggle**[1].

Preprocessing Steps:

Feature Engineering:

- Created new meaningful features to better capture traffic behavior:
  - **packets_per_sec:** Calculated as the number of source packets ('spkts') divided by the duration ('dur'). This feature highlights packet transmission intensity.

    - **bytes_per_sec:** Computed as the source bytes ('sbytes') divided by the duration ('dur'). This helps identify unusual data transmission patterns.

- Replaced infinite values resulting from division by zero (duration being zero) with NaN to avoid computational errors.

Label and Feature Separation:

- Clearly separated target labels and removed any potentially leaking features (labels or identifiers) from the feature set. Dropped columns included 'label', 'attack_cat', 'id', 'srcip', and 'dstip' to ensure unbiased model training.

Identifying Numeric and Categorical Features:

- Numeric features were selected automatically by detecting numeric data types.
- Categorical features were identified based on non-numeric data types.

Preprocessing Pipeline Construction:

Built a comprehensive preprocessing pipeline using ColumnTransformer:

- **Numeric Feature Pipeline:**
    - Applied SimpleImputer with a mean strategy to handle missing numeric values, ensuring no valuable numeric information was lost.
    - Scaled numeric features using StandardScaler to standardize value ranges, thereby improving the performance and stability of machine learning algorithms.
- **Categorical Feature Pipeline:**
    - Employed SimpleImputer with a 'most frequent' strategy to fill missing categorical values, preserving categorical distribution.
    - Transformed categorical features using OneHotEncoder with handle_unknown='ignore' to convert categorical data into numeric form without introducing bias from previously unseen categories.

Data Transformation:

- Applied the fitted preprocessing pipeline exclusively on the training set, ensuring no information leakage.
- Transformed the testing set using the same preprocessing steps defined and fitted on the training set, thus maintaining consistency in data processing between training and evaluation phases.

Output Verification:

- Verified the final shapes of preprocessed training and testing datasets to ensure successful execution and data readiness for modeling.

3. Methodology

Machine Learning Models:

- **Random Forest Classifier:**
  - Selected as a baseline model due to its ease of interpretation, robustness against noise, and effective performance in handling complex datasets.
  - Configured with 100 trees (n_estimators=100) and a fixed random state (random_state=42) for reproducibility.
- **XGBoost Classifier:**
  - Chosen as the final model for its high efficiency, superior performance, and strong capability to model feature interactions and handle imbalanced data.
  - Set with 100 trees (n_estimators=100), used label encoder suppression (use_label_encoder=False), evaluated using the 'logloss' metric (eval_metric='logloss'), and ensured reproducibility (random_state=42).

Training/Testing Details:

- Both models were trained using the previously preprocessed training dataset (X_train_processed).
- Predictions were generated on a distinct and consistently preprocessed testing dataset (X_test_processed) to fairly assess the generalization capabilities of the models.

- Consistent evaluation procedures and metrics were applied to allow direct comparison between models

4. Results & Analysis

Performance Metrics:

- Evaluated models comprehensively using multiple metrics including Accuracy, Precision, Recall, and F1-score, offering a balanced and insightful understanding of predictive performance.
- Utilized the classification_report function to obtain detailed precision, recall, and F1-score metrics per class, and accuracy_score to determine overall accuracy.
- Compiled all evaluation metrics into a clearly formatted table for ease of comparison and interpretation.

Visualizations:

- Generated confusion matrices using seaborn heatmaps for intuitive visualization of model performance:
    - **Random Forest Confusion Matrix:** Created using a blue color palette (cmap='Blues'), clearly depicting correct versus incorrect classifications.
    - **XGBoost Confusion Matrix:** Created using a green color palette (cmap='Greens') for clear visual differentiation and improved interpretability.
- These visualizations effectively illustrated the distribution of true positives, true negatives, false positives, and false negatives for each model.

Insights:

- Both models showed strong performance, achieving accuracy around 90%.
- XGBoost slightly outperformed Random Forest in key metrics, notably F1-score and recall, indicating its superior capability in accurately identifying network anomalies.
- Feature importance analysis (implicit within XGBoost) suggested that engineered features such as bytes per second and packets per second

played a significant role in accurately detecting anomalies, emphasizing the effectiveness of the feature engineering process.

5. Discussion & Challenges

Key Findings:

- XGBoost exhibited strong predictive capabilities, accurately identifying both common and subtle anomalies.
- Effective preprocessing significantly enhanced the accuracy and reliability of model outcomes.

Difficulties Faced:

- Initially faced issues with data leakage, which inflated model performance unrealistically.
- Encountered computational constraints due to the dataset's size and complexity.
- Handling missing and infinite values required careful attention to avoid bias or loss of information.

Potential Improvements:

- Further hyperparameter optimization might improve model accuracy.
- Implementing deep learning approaches such as neural network-based autoencoders could enhance anomaly detection capabilities.
- Additional feature engineering focusing on domain-specific knowledge could provide deeper insights and better model performance.

6. Conclusion & Future Work

Summary: This project demonstrated the successful application of machine learning algorithms for effective anomaly detection in network traffic, with XGBoost emerging as the superior model due to its high accuracy and balanced predictive capabilities.

Future Work:

- Investigate the effectiveness of deep learning models for real-time anomaly detection.

- Explore ensemble methods and hybrid approaches to further enhance detection rates.
- Extend the model application to more diverse cybersecurity scenarios and broader network environments.

7. GitHub Repository Link

Complete project code, datasets, preprocessing scripts, and comprehensive documentation are available at: https://github.com/zeinhord/network-anomaly-detection.git

The repository contains a detailed README file outlining the project structure, data preprocessing steps, model training and testing procedures, and instructions to replicate results.

8. Contributors

Ivan Leonychev T00727314:

- Data cleaning, preprocessing, and feature engineering.
- Model implementation and experimentation.
- Documentation and report writing.
- GitHub repository management and project presentation preparation.

Polina Kniazeva T00704993:

- Initial search and analysis
- Exploratory data analysis and initial data visualization.
- Model implementation and experimentation.
- Performance evaluation and visualization

9. References

- UNSW-NB15 Dataset: https://www.kaggle.com/datasets/mrwellsdavid/unsw-nb15/data