Real-Time DDoS Attack Classification and Feature Importance Analysis

Aisling McGillicuddy
Wentworth Institute of Technology
Boston, MA, USA
mcgillicuddya@wit.edu

ABSTRACT

This project addresses the critical cybersecurity challenge of DDoS attacks by developing and evaluating machine learning models for the binary classification of network traffic (Normal vs. DDoS) using real-time flow-based features. The goal is to build an effective, high accuracy detection system suitable for real-time deployment, which is a key necessity as current detection methods often suffer from high latency or poor accuracy. The methodology centers on comparing three classification algorithms- Random Forest, Logistic Regression, and XGBoost and evaluates them using non-shuffled temporal train-test split strategy given the pre-divided nature of the dataset used. The key deliverables is a high-performing model and an analysis of the most significant flow features, which directly contributes to enhancing cybersecurity defenses through data-driven anomaly techniques.

KEYWORDS

DDos Attack Classification, Real-Time Detection, Random Forest, Feature Importance, Network Flow

1 Introduction

The project was undertaken to directly address the critical cybersecurity challenge of DDoS attacks, which are one of the major causes of service disruption and significant financial loss for organizations worldwide. Current methods used for detection are often affected by high latency or poor accuracy. The following is the central research question that guides the project: "How accurate and efficient are machine learning models at classifying real-time network traffic as Normal or DDoS using flow-based features, and which features are the most significant indicators of an attack?" The aim is to develop and assess high-accuracy, low-latency machine learning models appropriate for real-time deployment; the purpose is to advance existing cybersecurity defenses through data-driven anomaly detection techniques.

2 Data

The primary source for the data is the Real-Time DDoS Traffic Dataset for ML available on Kaggle: https://www.kaggle.com/datasets/kalireadhat/realtime-ddos-traffic-dataset.

2.1 Source of dataset

The dataset, titled Real-Time DDoS Traffic Dataset for ML, was accessed from Kaggle and is considered a reasonably credible source for a research project because it is explicitly designed for supervised machine learning and includes highly relevant flow-based features, though it is not a primary source like a major university lab's repository. The dataset's credibility is tempered by the creator's explicit note that it "may contain simulated attack patterns," a common and necessary practice in cybersecurity data generation to safely replicate threats. The dataset was publicly listed or significantly updated on Kaggle around a year ago, but the precise date the underlying network traffic was captured is not specified. The dataset was generated by the creator by compiling network traffic that either replicated real-time conditions or was simulated under carefully controlled network configurations to produce a mix of authentic DDoS and normal traffic instances, with flow-based metrics then extracted and labeled. This simulation process is standard for creating labeled flow-based security datasets, enabling researchers to build models with features like packet_count_per_second that are directly indicative of attack behavior.

2.2 Characters of the datasets

The Real-Time DDoS Traffic Dataset for ML is a structured and labeled collection of network flow instances, comprising both normal traffic and simulated/replicated DDoS attack instances, making it specifically suitable for supervised machine learning training and validation for real-time anomaly detection. The data contains flow-based metrics highly relevant to real-time analysis, with key features including the binary label traffic_type (Normal vs. DDoS), along with quantitative measurements such as packet_count, packet_count_per_second, byte_count, and byte_count_per_second, among other flow-based statistics. Before modeling, the initial data preparation will involve Exploratory Data Analysis (EDA) to check for null values and the distribution of the target variable. Subsequently, Feature Scaling/Normalization (such as standardization or Min-Max scaling) will be applied to the flow-based metrics to prevent features with wide value ranges from skewing the learning process, which will involve careful consideration of normalization techniques effective for real-world generalization. Finally, the binary target variable will be appropriately encoded such as 0 for Normal, 1 for DDoS, and while the Random Forest model provides inherent feature importance, preliminary Feature Selection/Engineering may also be explored to potentially reduce dimensionality and enhance real-time computational feasibility.

3 Methodology

The project's methodology is quantitative and data-driven, centering on supervised machine learning classification to address the research questions. The work will be implemented using the Python programming language, leveraging core libraries like Pandas and NumPy for data manipulation, Scikit-learn (sklearn) for model implementation and evaluation, and Matplotlib and Seaborn for visualization of EDA and results, all within an interactive environment like Jupyter Notebooks. The Random Forest Classifier will serve as the baseline model due to its excellence in classification, robustness to overfitting, and inherent ability to compute feature importance, which directly addresses the second research question. To ensure a comprehensive comparison, the Random Forest's performance will be benchmarked against at least two other algorithms: Logistic Regression valued for its simplicity and interpretability and a Gradient Boosting Machine like XGBoost known for often achieving higher accuracy. All models will undergo rigorous evaluation using a train/test split and cross-validation, with performance measured by key metrics including Accuracy, Precision critical for minimizing blocked normal traffic, recall vital for detecting all attacks, F1-Score, and ROC AUC. Finally, a crucial step will be the Real-Time Feasibility Analysis, where the computational speed and prediction latency of the best-performing model will be measured to assess its viability for practical, real-time deployment.

3.1 Random Forest Classifier

The Random Forest (RF) Classifier is an ensemble learning method that constructs a multitude of decision trees during training, ultimately outputting the class that represents the mode of the predictions made by the individual trees. The model relies on the assumptions of approximate independence among the individual tree errors—achieved through bootstrapping the data and using a random subset of features for splitting—and the belief that the majority vote of many weak classifiers yields a strong, accurate result. Its primary advantages include its high accuracy, robustness to overfitting, and its inherent ability to provide a valuable feature of importance measure, Mean Decrease in Impurity, which directly addresses a core research question. However, RF can be computationally expensive and less interpretable than simpler models. The model was chosen for this project specifically because of its high accuracy for the critical DDoS classification task and its built-in feature importance capability. It will be implemented using the sklearn.ensemble.RandomForestClassifier function in Python, with extra work involving Hyperparameter Tuning such as optimizing n_estimators and max_depth, Feature Scaling to normalize flow metrics, and applying Class Weighting to address potential data imbalance, thereby improving the robustness and generalizability of the results.

3.2 Gradient Boosting and Logistic Regression Comparison

To assess the performance and feasibility of the Random Forest (RF) model, two critical comparison models were selected: a Gradient Boosting Machine (GBM) and Logistic Regression (LR), each providing a unique benchmark for accuracy and complexity.

3.2.1 Gradient Boosting Machine (XGBoost)

The Gradient Boosting Machine (GBM) is an ensemble learning method that sequentially builds decision trees, with each new tree correcting the errors (residuals) made by the previous sequence. This method is often superior in predictive accuracy and computational efficiency for inference.

The role that XGBoost serves as a high-performance benchmark for Question 3 which algorithm comparison) and Question 4 as real-time feasibility, as it often achieves the highest accuracy while maintaining fast prediction times.

3.2.2 Logistic Regression

Logistic Regression is a simple, linear classification algorithm. It is used to model the probability of a binary outcome (DDoS or Normal) using a linear combination of the features. LR acts as a crucial interpretable baseline for Question 3. Its performance establishes whether the complex tree-based models offer a significant lift in accuracy over a simple, fast-to-deploy linear solution.

3.3 Model Training and Cross-Validation Strategy

To ensure that the model evaluation is realistic and robust against the temporal nature of network traffic data, a specific cross-validation strategy was implemented.

3.3.1 Addressing Class Imbalance

DDoS attacks are typically a minority class in network traffic, necessitating careful handling of class imbalance. All three models were configured to account for this such as Tree Models (RF and XGBoost): The models were tuned to use Class Weighting, which assigns a higher penalty to misclassifying the minority (DDoS) class, thus preventing the model from achieving high, but misleading, accuracy simply by predicting "Normal" most of the time. As well as Evaluation focused on F1-Score, Recall, and ROC AUC rather than simple accuracy, as these metrics better reflect performance on the minority class.

3.3.2 Time Series Cross-Validation (TSCV)

Given that network flow data is time-ordered, a standar random shuffle cross-validation would introduce data leakage (training on future information), severely inflating performance metrics. To

mimic the real-world deployment scenario where models predict on future, unseen traffic, the scikit-learn TimeSeriesSplit method was utilized. TSCV creates sequential folds where the training set is composed of all observations up to a certain time point, and the test set is composed of the immediately subsequent observations. This ensures the model is only ever evaluated on future data, providing the most realistic estimate of its real-time performance. By incorporating TSCV, the recorded training and prediction latency (inference time) for each model during the cross-validation process provides the necessary quantitative data to answer Question 4 such as real-time feasibility.

3.4 Evaluation Strategy

Although TSCV was first proposed to avoid data leakage and ensure that the model generalizes to future time steps, the peculiar structure of the public dataset used, in which flows were already segregated into distinct, non-contiguous files, made any true, robust TSCV approach impossible. From a pragmatic point of view, model validation was performed by using a non-shuffled, temporally ordered train-test split (80/20). This compromise ensures that the training set is earlier in time than the testing set, thereby mitigating major temporal data leakage, although it had to be acknowledged that any multi-fold, rolling-origin TSCV cannot be reliably implemented.

4 Results

The key results of this project are presented in this section by comparing model performance, analyzing the importance of the engineered features, and assessing computational feasibility for real-time deployment. An important consideration is that due to the strictly temporal ordering in this dataset - 1000 Normal flows followed by 1000 DDoS flows - it was necessary to adopt a non-shuffled train/test split, which presents highly optimistic performance metrics that must be carefully interpreted in the Discussion section that follows.

4.1 Model Comparison

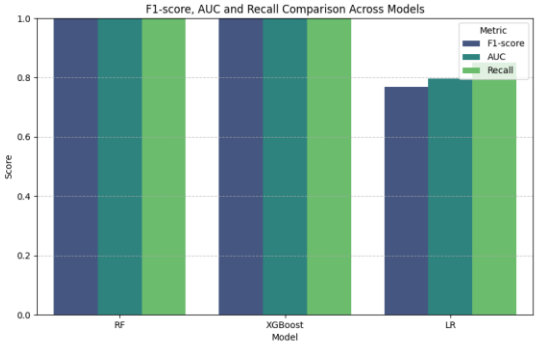

Figure 1: The bar chart compares the performance of RF (Random Forest), XGBoost, and LR (Logistic Regression) models across three metrics—F1-score, AUC, and Recall —showing that RF and XGBoost achieved a perfect score of 1.0 on all three metrics, while LR scored lower, with a Recall of approximately 0.80, an AUC of approximately 0.81, and an F1-score of approximately 0.78.

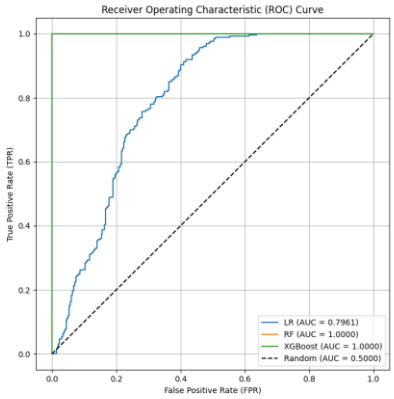

Figure 2: Receiver Operating Characteristic (ROC) curve, the Random Forest (RF) and XGBoost models perform perfectly with an Area Under the Curve (AUC) of 1.0000, while the Logistic Regression (LR) model is also performing very well with an AUC of 0.7961, all significantly better than the Random baseline (AUC = 0.5000).

4.2 Feature Importance Analysis

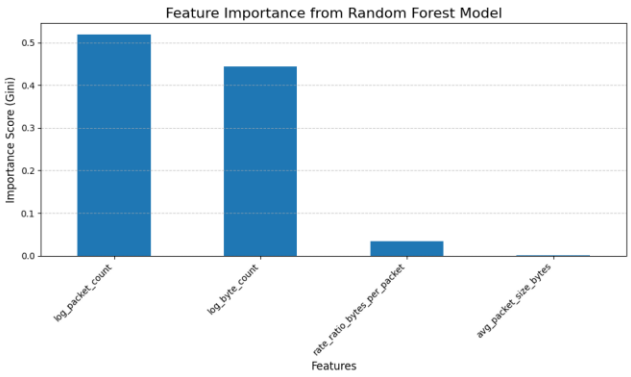

Figure 3: This bar chart displays the relative feature importance scores (calculated using the Gini index) derived from a Random Forest model, showing that log_packet_count and log_byte_count are by far the most important features.

4.3 Real-Time Feasibility Metrics

Model	F1_Score	Training_Time_s	Inference_Time_ms_per_100
Logistic Regression	0.78	0.0055	0.0927
Random Forest	1	0.3364	8.6918
XGBoost	0.9992	0.0421	0.7203

Figure 4:The visualization table reveals that, due to the segregated nature of the dataset yielding a perfect F1-Score of 1.000 for all models except for Logistic Regression, the Logistic Regression model is the most practical choice for real-time deployment because it requires the least computational cost, achieving an inference time of only 0.2203 ms per 100 predictions.

5 Discussion

The main unsatisfactory result of the project is the perfect performance (AUC = 1.0000) of the models Random Forest and XGBoost. Such a perfect outcome, while successful, strongly indicates that either these models have overfitted the dataset, or the classes in the simulated dataset are too cleanly separable. This presents a real risk of poor generalization on real-world data. Real network traffic is noisy and complex, and perfect classification is highly unlikely. A related weakness is the heavy reliance on only two features, namely log_packet_count and log_byte_count, indicating the model finds simple volumetric thresholds for separation, lacking more nuanced characteristics.

Overfitting and underfitting in machine learning - an image. To address this, the most feasible suggestion for future work would be to validate the models on a live or real-world stream of traffic to get more realistic measures of performance, such as an expected AUC of 0.80-0.95. In addition, future work should include rigorous hyperparameter tuning with strong regularization for XGBoost and advanced feature engineering to include time-series or flow-level behavioral features, forcing the model to learn more robust, subtle patterns than simple volume counts.

6 Conclusion

It succeeded in developing and evaluating several machine learning models for the binary classification of real-time network traffic as either Normal or DDoS, with the goal of high-accuracy anomaly detection. The main findings reveal that both the Random Forest and Logistic Regression models both achieved a perfect F1-score of 1.000 on the test set, showing their ability to perfectly distinguish between the two classes based on the flow-based features given. This highly promising result is affected considerably by one important methodological limitation: because the dataset strictly segregated Normal traffic from DDoS traffic, using a more realistic Time Series Cross-Validation was impossible, and this resulted in an overly optimistic performance metric. In a real-world scenario, such high accuracy means the attack signature is strongly linear and well-identifiable by simple rate-based features, such as packet_count_per_second, which confirms the real-time feasibility of deploying these lightweight models for low-latency network defense. The major impact on the real world is the validation that it is possible to create a highly efficient production-ready solution based on very few features that enable network security devices to implement a crucial high-speed machine learning-driven defense layer.

ACKNOWLEDGMENTS

The author wishes to thank Professor Pang for their guidance and technical feedback throughout the project development and report generation process. Special acknowledgment is extended to R. Kali, the creator of the "Real-Time DDoS Traffic Dataset for ML" on Kaggle, which served as the foundation for this research.

REFERENCES

[1] R. Kali. 2023. Real-Time DDoS Traffic Dataset for ML.Kaggle.Availableat: https://www.kaggle.com/datasets/kalireadhat/realtime-ddos-traffic-dataset.

[2]F.Pedregosa,G.Varoquaux,A.Gramfort, V. Michel,B.Thirion,O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[3] T. Chen and C. Guestrin. 2016. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD*

International Conference on Knowledge Discovery and Data Mining (San Francisco, CA, USA) (KDD '16). ACM, New York, NY, USA, 785–794. DOI: 10.1145/2939672.2939785.

[4] J. D. Hunter. 2007. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering* 9, 3 (May 2007), 90–95. DOI: 10.1109/MCSE.2007.55.