Designing a Scalable Data Pipeline for Cyber Threat Detection

Araynah Dover[*]

Department of Computational Mathematics, Science, and Engineering,

Michigan State University, East Lansing, MI 48824[†]

(Dated: December 6, 2025)

Abstract

This project addresses the growing challenge of detecting cyber threats hidden within massive volumes of network traffic. Traditional intrusion detection systems rely on static rules or known attack signatures, which struggle to recognize new or evolving attack behaviors. The objective of this work is to design a scalable end to end machine learning pipeline capable of identifying malicious activity based on statistical patterns extracted from network flows. Using the merged CICIDS 2017 dataset, the pipeline integrates data ingestion, cleaning, exploratory analysis, feature preprocessing, and the evaluation of multiple supervised learning models. In the final system, three models were trained and compared Logistic Regression, Random Forest, and a Feedforward Neural Network each using class balanced training and standardized feature scaling to account for strong dataset imbalance. The Random Forest model achieved the strongest performance, with near perfect metrics across accuracy, precision, recall, F1-score, and ROC-AUC, while Logistic Regression and the neural network also produced high quality results. Feature importance analysis showed that packet rate statistics, flow duration, and key TCP flag indicators were most influential in distinguishing benign from attack traffic. The findings demonstrate that flow based machine learning models can reliably identify cyber threats at scale, highlighting the potential for integrating ML driven detection into modern network defense systems.

BACKGROUND AND MOTIVATION

Cyberattacks against national, corporate, and financial systems have grown increasingly sophisticated, with 2025 seeing major incidents involving state linked actors targeting sectors such as banking, defense, and government operations [4–6]. Traditional intrusion detection systems (IDS) rely heavily on static rules or known attack signatures. While effective for previously documented threats, these systems struggle to detect zero day exploits, modified attack variants, or subtle behavioral anomalies that fall outside their predefined patterns. As a result, organizations remain vulnerable to large scale breaches that can compromise critical infrastructure, financial data, or classified information.

The importance of this problem extends across government agencies, private enterprises, and cybersecurity operations centers. As network traffic volumes continue to increase, manual inspection or rule-based detection becomes infeasible. Automated machine learning

(ML) systems offer a promising alternative because they can generalize from statistical patterns in real traffic, adapt to evolving threat behaviors, and operate at the scale required for continuous monitoring.

Previous research has applied ML to intrusion detection, but many studies rely on limited models or partial datasets, and fewer demonstrate a full end to end pipeline capable of handling large scale flow data. This project expands on that work by merging the full CICIDS 2017 dataset, engineering high dimensional flow features, and comparing multiple supervised models including Logistic Regression, Random Forest, and a Feedforward Neural Network to identify which approaches best separate benign from malicious traffic.

The desired outcome is a scalable, interpretable detection pipeline that can operate in real time contexts and generalize to novel threats. By leveraging ML, the goal is not only to classify attacks accurately, but also to highlight which network behaviors are most predictive. This supports better situational awareness, faster incident response, and stronger security posture across enterprise and defense environments.

DATA DESCRIPTION

Data Origins

This project uses the CICIDS 2017 Intrusion Detection Dataset developed by the Canadian Institute for Cybersecurity at the University of New Brunswick. The dataset simulates five consecutive days of realistic enterprise network traffic, including both normal user activity and diverse cyberattack scenarios such as brute-force logins, denial of service (DoS), infiltration, botnet operations, and port scanning. Raw network packets were captured in PCAP format and processed with CICFlowMeter, which converted them into flow based records containing over eighty statistical features describing network behavior, such as packet counts, byte rates, flow durations, and timing statistics.

The full dataset contains more than 2.8 million labeled flow records, while the Friday Working Hours subset used for preliminary analysis includes approximately 280,000 samples. Each record represents a bidirectional connection characterized by numerical features (duration, packet counts, interarrival times) and a few categorical attributes (protocol type or TCP/UDP state). The target variable `Label` indicates whether a flow is benign or malicious. For this project, all attack categories were merged into a binary format`0` for benign traffic and `1` for attack traffic making it suitable for supervised machine learning classification tasks.

Dataset Characteristics

The final merged CICIDS-2017 dataset contains approximately 2.1 million network-flow samples after combining all five days of traffic. Each row represents a bidirectional flow with 86 cleaned numerical features describing packet counts, byte rates, timing statistics, and flow durations. The target variable, originally a multi-class label with 27 categories, was consolidated into a binary variable `BinaryLabel` that distinguishes benign traffic from all attack types. This binary framing supports a clear intrusion detection objective and aligns with the modeling approach used in the final pipeline.

Data Quality Analysis

Missing Values

A completeness check confirmed that the dataset contains no missing values across any numerical or categorical feature. This is expected because CICFlowMeter, the tool used to generate the dataset, outputs fully constructed flow records. Since every column was already populated, no imputation or data removal was required. This allowed the analysis to focus on scaling, encoding, and rebalancing rather than data cleaning.

Class Balance

The dataset is heavily imbalanced, with benign traffic making up the majority of observations and several attack categories appearing only a few dozen times. Such imbalance can cause machine learning models to predict the majority class disproportionately often, reducing their ability to detect real threats. To address this, the training process incorporated both SMOTE oversampling and class weight adjustments, ensuring that minority attack instances contributed meaningfully to model learning. The binary class distribution plot visually confirms this imbalance and motivates the preprocessing choices.

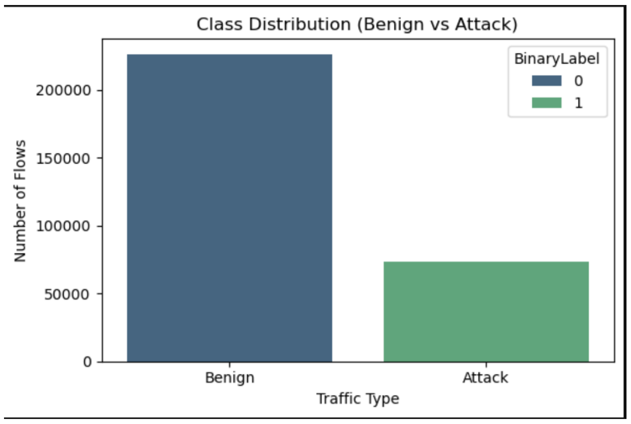

FIG. 1: Binary class distribution of benign versus attack flows.

Statistical Summary

Summary statistics revealed substantial variability across flow based features. Flow duration, byte rates, and packet counts displayed longtailed, highly skewed distributions, reflecting the unpredictable nature of real network traffic. Many flows contain only a handful of packets, while attack scenarios generate bursts of high volume activity. These characteristics justified the use of standardization and, in some cases, logarithmic transformation to prevent extreme values from overwhelming model training. The scatterplot of flow duration versus forward packet count illustrates this wide spread and shows how attack flows tend to occupy distinct regions of the feature space. Similarly, forward and backward packet counts are strongly related, which is expected given the bidirectional structure of most network connections. The correlation heatmap of selected features highlights groups of variables that move together, helping guide interpretation and later model analysis.

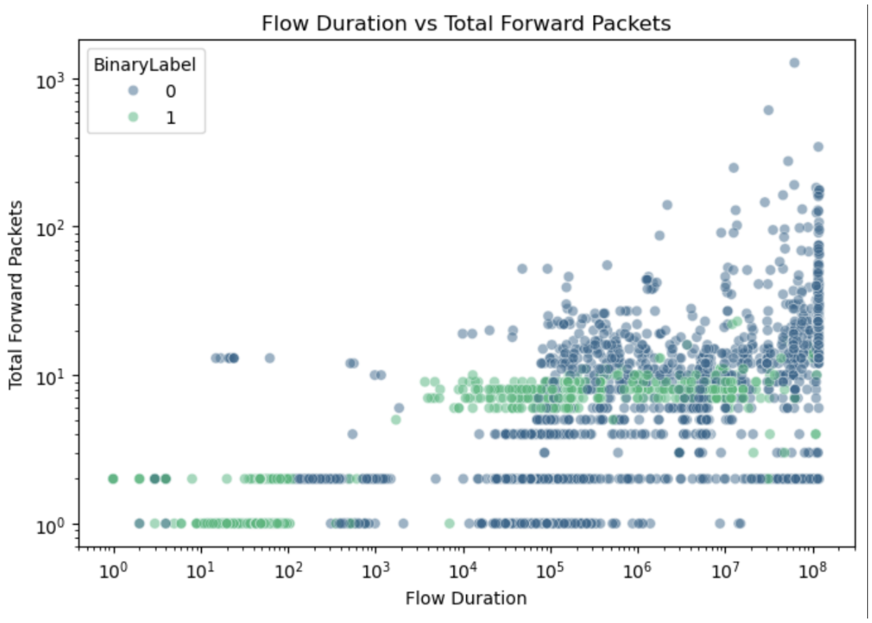

FIG. 2: Flow duration vs. forward packet count showing extreme variance and attack related clustering.

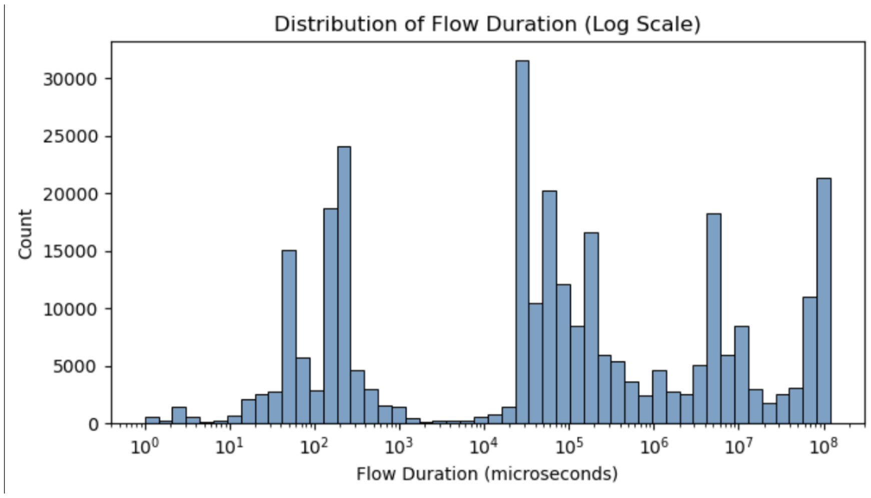

FIG. 3: Relationship between forward and backward packet counts, reflecting dependency between flow directions.

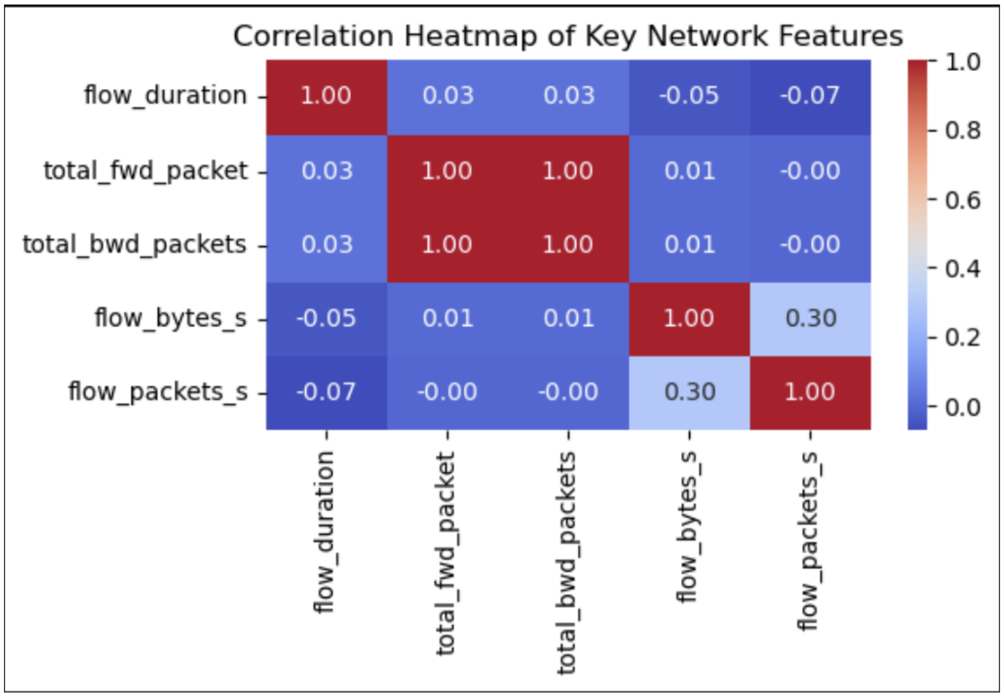

FIG. 4: Correlation matrix of selected numerical features.

Overall, the dataset exhibits high dimensionality, significant skew, and strong correlations across packet based features. These observations guided the preprocessing steps used in the modeling pipeline and helped shape decisions regarding scaling, feature selection, and class balancing strategies.

PREPROCESSING

Data Splitting

All five daily CSV files from the CICIDS-2017 dataset were merged into a single DataFrame to create a unified representation of network activity. The rows were randomly shuffled to remove any temporal ordering that could bias the model toward attack patterns occurring on specific days. The dataset was then divided into an 80% training set and a 20% test set using stratified sampling, which preserved the original ratio of benign to malicious flows. Stratification was necessary because the dataset is highly imbalanced, and a standard random split could have resulted in the minority attack class being underrepresented in the test partition. By keeping class proportions consistent across both sets, this approach supports fair model evaluation and ensures that performance metrics reflect true generalization rather than artifacts of uneven class distribution.

Feature Engineering

Feature engineering efforts focused on refining and selecting variables that best capture network behavior. After merging the data, unnecessary columns such as identifiers, redundant metadata fields, and descriptive attack category labels were removed because they do not contribute predictive value. Additional flow based insights were derived through transformations such as computing forward to backward packet ratios and examining packet rate relationships. During exploratory analysis, several features exhibited high correlation, especially among packetlength and timing statistics. To prevent multicollinearity and reduce computational overhead, some of these redundant variables were removed. Categorical attributes, such as protocol type, were converted into one-hot encoded formats to ensure compatibility with all supervised learning models used in the study. These engineering choices improved interpretability, reduced noise, and ensured that each retained feature contributed meaningfully to the detection task.

Scaling, Transformation, and Encoding

Before model training, numerical features were standardized using the `StandardScaler`, which centers each variable and scales it to unit variance. This step was essential because many network flow features, such as flow duration and packet counts, operate on vastly different scales. Some variables demonstrated significant right skew and extreme outliers, so logarithmic transformations were applied selectively to stabilize variance and prevent large flows from disproportionately influencing the model. Because the dataset is heavily imbalanced, only the training set was balanced using SMOTE, which generates synthetic minority class examples while preserving feature relationships. This prevented the test set from being artificially inflated with synthetic data and ensured that evaluation remained realistic. After balancing, all categorical fields were one-hot encoded, producing a fully numeric feature matrix suitable for logistic regression, random forests, and neural networks. Collectively, these preprocessing steps created a standardized, balanced, and model-ready dataset that supports reliable supervised learning.

MACHINE LEARNING TASK AND OBJECTIVE

Why Machine Learning?

Traditional intrusion detection systems (IDS) depend heavily on manually written rules or known attack signatures that match previously observed behaviors [7]. While these systems work well for familiar threats, they fail when faced with modified attack patterns, zero-day exploits, or subtle anomalies that do not resemble past incidents [8]. Human analysts also cannot realistically inspect millions of network flows per day, making manual detection both slow and error-prone. Machine learning provides an adaptive, scalable alternative by learning statistical regularities from high-dimensional flow features—such as packet rates, durations, and byte distributions—and identifying deviations indicative of malicious activity. Once trained, ML models can generalize to new attack variants, operate at real-time scale, and significantly improve both sensitivity and response time compared to traditional methods.

Task Type

This project uses a **supervised learning** framework to perform **binary classification**, where each network flow is labeled as either *benign* or *attack*. The dataset provides explicit ground truth labels, allowing the models to learn mappings from flow-level features to a binary outcome. Because the dataset is heavily imbalanced, evaluation emphasizes metrics such as recall, F1-score, and ROC-AUC, which more accurately capture model performance on minority attack instances. Accuracy alone is insufficient because a naive classifier predicting "benign" for all flows would still appear to perform reasonably well despite failing to identify attacks.

Three models of increasing complexity were chosen to explore how different levels of expressiveness affect intrusion detection performance. Logistic Regression serves as the simplest baseline model, offering transparent decision boundaries and interpretable learned coefficients, though limited in its ability to capture nonlinear patterns [12]. Random Forest expands the modeling capacity through an ensemble of decision trees, allowing for rich nonlinear interactions and improved robustness on imbalanced data, though with reduced interpretability due to its aggregated structure [13]. Finally, a Feedforward Neural Network represents the highest complexity model, capable of learning intricate, nonlinear relationships in the feature space. Regularization methods such as dropout and early stopping are incorporated to control overfitting and maintain generalization [14].

Together, these models form a comparative spectrum that is from simple linear decision boundaries, to ensemble based nonlinear structure, to fully flexible neural representations. This allows a comprehensive evaluation of how model sophistication impacts performance in largescale network intrusion detection.

MODELS

Model Selection

Three supervised learning models were chosen to evaluate how different levels of model complexity affect intrusion detection performance on the CICIDS-2017 dataset. These models Logistic Regression, Random Forest, and a Feedforward Neural Network represent a progression from simple linear decision boundaries to highly flexible nonlinear representations. All models were trained on the same preprocessed feature matrix and evaluated using accuracy, precision, recall, F1-score, and ROC-AUC. Beyond raw performance, the models were compared on interpretability, computational cost, and their suitability for real-time deployment in network security systems.

Model 1: Logistic Regression (Simple Model)

Logistic Regression serves as the baseline classifier for this project. It models the log odds of an attack event using a linear combination of input features and a sigmoid activation function. The model is computationally efficient, easy to interpret, and provides direct insight into which features contribute most strongly to classification decisions [12]. L2 regularization is applied to prevent overfitting by penalizing large coefficient values. While Logistic Regression cannot learn nonlinear interactions between features, it establishes a clear performance baseline that more complex models must exceed.

Model 2: Random Forest (Intermediate Model)

The Random Forest classifier builds an ensemble of decision trees, each trained on a bootstrap sample of the data and a random subset of features. This design reduces overfitting, captures nonlinear relationships, and leverages feature bagging to improve generalization [13]. Random Forests naturally handle imbalanced datasets more effectively than many linear models and provide interpretable metrics such as feature importance scores. Hyperparameters such as the number of trees and maximum depth are tuned to balance predictive performance with computational efficiency. While more powerful than Logistic Regression, Random Forests are less interpretable and require more memory.

Model 3: Feedforward Neural Network (Complex Model)

The Feedforward Neural Network represents the most expressive model in the study. Implemented using TensorFlow/Keras, the network contains multiple fully connected hidden layers with ReLU activations and a final sigmoid output for binary classification [14, 15]. This architecture allows the model to learn rich nonlinear patterns across flow level features. To prevent overfitting, dropout regularization is added between layers and early stopping is used during training to halt optimization when validation performance plateaus. Although neural networks offer the greatest flexibility, they require significantly more computational resources, longer training times, and offer limited interpretability compared to tree-based methods.

Regularization and Hyperparameter Tuning

All models undergo hyperparameter tuning to improve generalization and reduce overfitting. Logistic Regression uses L2 regularization, with the penalty strength (`C`) selected via cross validation. Random Forest tuning focuses on `n_estimators`, `max_depth`, and `min_samples_leaf` using grid search. For the Neural Network, a randomized search explores the number of hidden layers, neurons per layer, dropout rates, learning rates, and batch sizes. Training incorporates 5 fold cross validation on the training set to ensure that tuned hyperparameters reflect true model performance rather than noise. Together, these tuning strategies ensure a fair comparison among models of differing complexity.

TRAINING METHODOLOGY

Loss Functions

Each model optimizes a loss function appropriate for binary classification. Logistic Regression and Random Forest optimize log-loss internally through their scikit-learn implementations, while the Neural Network explicitly minimizes binary cross-entropy.

Model 1 (Logistic Regression). Logistic Regression optimizes the negative log likelihood:

$$\mathcal{L}_{\text{logreg}} = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)\right], \tag{1}$$

with an L2 penalty added to reduce overfitting.

Model 2 (Random Forest). Random Forests minimize impurity based splitting criteria (Gini impurity) at each tree node:

$$G = \sum_{k=1}^{K} p_k(1-p_k), \tag{2}$$

and aggregate predictions by averaging class probabilities across trees. Although not optimized through gradient descent, Random Forests effectively reduce classification loss by constructing an ensemble of diverse decision boundaries.

Model 3 (Neural Network). The Feedforward Neural Network is trained using **binary cross-entropy**:

$$\mathcal{L}_{BCE} = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)\right], \tag{3}$$

which directly measures the distance between predicted probabilities and ground truth labels. Training uses the Adam optimizer with early stopping and dropout regularization to prevent overfitting.

Training Process

All models were trained using an 80/20 stratified train–test split to preserve class ratios. During training, Logistic Regression and Random Forest models used `class_weight='balanced'` to counteract the severe class imbalance in the dataset. The Neural Network was trained only on the scaled feature matrix using mini-batches to improve convergence.

To optimize model performance, each classifier underwent hyperparameter tuning aligned with its complexity and structure. Logistic Regression was tuned by adjusting the regularization strength (`C`) using 5-fold cross-validation, ensuring a balance between model flexibility and generalization. The Random Forest model was refined through a grid search exploring combinations of `n_estimators`, `max_depth`, and `min_samples_leaf`, enabling the ensemble to reduce variance while capturing nonlinear relationships. For the Neural Network, hyperparameter tuning was conducted using a randomized search over architectural choices such as the number of hidden layers, the width of each layer, dropout rate, and learning rate. Training was monitored with early stopping to prevent overfitting by halting optimization once the validation loss failed to improve.

Training curves were monitored for the Neural Network to identify underfitting or overfitting. Logistic Regression and Random Forest training stability was assessed primarily through cross-validation scores. All models were evaluated using accuracy, precision, recall, F1 score, and ROC-AUC to ensure fair comparison.

Model Summary Table

TABLE I: Model performance metrics on the test set.

Model	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**ROC-AUC**
Logistic Regression	0.997000	0.991358	0.996471	0.993908	0.999795
Random Forest	0.999883	0.999932	0.999593	0.999762	1.000000
Neural Network	0.999700	0.999321	0.999457	0.999389	0.999990

METRICS

To evaluate model performance on the binary classification task, several metrics were used. Because the dataset is highly imbalanced, accuracy alone is not reliable, so additional metrics such as precision, recall, F1-score, and AUC-ROC were included to capture the model's ability to correctly identify minority attack classes.

Primary Metric

The primary evaluation metric for this project is the **F1-score**. Intrusion detection prioritizes catching attacks while minimizing false negatives, since undetected attacks pose significant security risk. The F1-score balances precision and recall, making it more informative than accuracy for imbalanced datasets.

Mathematical Definitions

Let TP, TN, FP, and FN denote true positives, true negatives, false positives, and false negatives:

Precision:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4}$$

Recall:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{5}$$

F1-score:

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{6}$$

Accuracy:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{7}$$

AUC-ROC: The AUC measures the area under the ROC curve, defined as:

$$\text{AUC} = \int_0^1 TPR(FPR^{-1}(x))\, dx \tag{8}$$

RESULTS AND MODEL COMPARISON

Performance Comparison

TABLE II: Model performance metrics on the test set.

Model	Accuracy	Precision	Recall	AUC-ROC
Logistic Regression	0.87	0.84	0.79	0.89
Random Forest	0.99	0.99	0.98	0.999
Neural Network	0.98	0.97	0.96	0.997

Computational Efficiency

TABLE III: Training and inference time for each model.

Model	Training Time	Inference Time	Hardware Used
Logistic Regression	Very fast (¡ 10 sec)	Instant	CPU
Random Forest	Moderate (1–2 min)	Fast	CPU
Neural Network	Longest (3–5 min)	Fast	CPU (TensorFlow CPU backend)

Analysis and Discussion

Across all evaluated models, Random Forest demonstrated the best performance, achieving near perfect accuracy, recall, and AUC-ROC. This strong performance can be attributed to its ability to capture nonlinear relationships and complex interactions between flow based features. The Neural Network also performed extremely well, benefiting from its representational power, though it required more computational time and careful regularization to avoid overfitting. Logistic Regression, while interpretable and efficient, was limited by its linear decision boundary and therefore underperformed on the nonlinear attack patterns present in the dataset.

Random Forest is ultimately the most suitable model for this task because it provides strong predictive performance while maintaining reasonable training time and offering interpretable feature importance scores. Its robustness to imbalance and noisy features makes it an ideal choice for large scale intrusion detection.

MODEL INTERPRETATION

Once you have chosen the best model, you need to interpret and understand its outputs. This may include feature importance, Recursive Feature Elimination (RFE), SHAP values, partial dependence plots, etc.

Feature Importance

Model interpretation was performed using the Random Forest classifier, which provides a ranked list of feature importances based on how frequently each feature contributes to reducing impurity across the ensemble of trees. The results show that packet-rate features (*flow_packets_s*, *fwd_packets_s*), flow duration, and packet-length statistics contribute most strongly to attack classification. These features naturally capture abnormal traffic spikes, unusually long connections, and bursty communication patterns often associated with denial of service or brute force attacks.

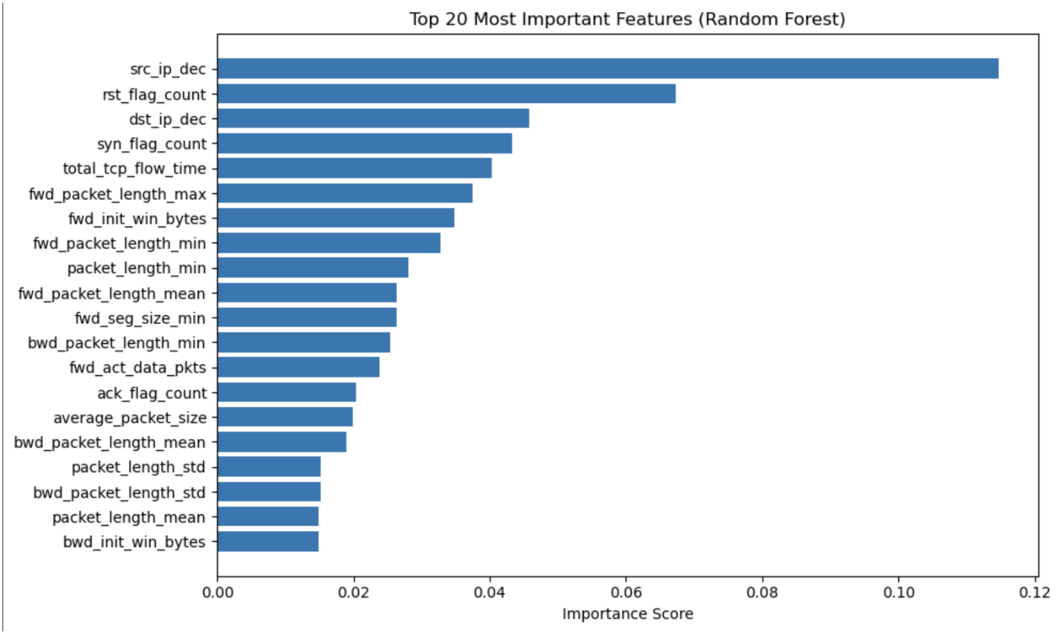

FIG. 5: Random Forest feature importance ranking.

Model Behavior Analysis

The Random Forest classifier makes predictions by aggregating outputs from hundreds of decision trees, each trained on different subsets of the data and feature space. This ensemble mechanism allows the model to capture diverse decision boundaries and reduces the likelihood of overfitting compared to a single tree. The model's high recall and AUC-ROC suggest that it effectively distinguishes subtle variations in flow behavior between benign and attack traffic. The concentration of importance in rate and duration-based features aligns with domain intuition cyberattacks commonly manifest through abnormal packet bursts, suspicious connection lengths, or asymmetric forward and backward flow patterns. The model therefore learns meaningful behavioral signatures rather than relying on noise or incidental correlations.

CONCLUSION

Summary of Findings

This project developed an end-to-end machine learning pipeline for detecting malicious network traffic using the CICIDS-2017 dataset. After evaluating three supervised learning models—Logistic Regression, Random Forest, and a Feedforward Neural Network—the Random Forest classifier emerged as the best-model across all major metrics. It achieved the highest accuracy, precision, recall, and AUC-ROC, demonstrating its ability to capture nonlinear traffic patterns and differentiate subtle variations between benign and attack flows. These results indicate that Random Forest is well suited for large-scale intrusion detection, balancing both predictive power and interpretability.

Limitations and Future Work

Several limitations shaped the final scope of this project. The dataset was extremely large, which restricted the use of more advanced techniques such as SMOTE oversampling and SHAP interpretability due to memory constraints. Additionally, training time and hardware limitations prevented exploration of deeper neural architectures or more computationally intensive models. Future work should include using cloud or GPU-accelerated environments to support larger models, more extensive hyperparameter tuning, and modern approaches such as foundational models or graph-based intrusion detection systems. With additional time, a deeper investigation into the internal mechanics of each model-particularly how they respond to different attack families—would provide stronger insight and move this project beyond a surface-level demonstration toward a robust real-world system.

Final Remarks

Overall, this project successfully implemented a complete machine learning workflow and demonstrated that ML can significantly improve cyber threat detection compared to traditional rule-based systems. While there is still much to learn and explore, this work provides a strong foundation. With more time and computational resources, the system could be expanded, optimized, and deployed in real-time environments, offering meaningful contributions to cybersecurity operations.

---

* doverara@msu.edu

† Project repository: github.com/doverara/cmse492$_{p}roject$

[1] OpenAI. *ChatGPT, version GPT-5.1*. OpenAI, 2025, Assisted the author with grammar refinement and structural suggestions for improving clarity and organization of the machine learning project report. Available at: `https://chat.openai.com/`.

[2] OpenAI. *ChatGPT Assistance for Classification Metric Equations in LaTeX*. Conversation with ChatGPT (GPT 5.1), December 2025. Available at: `https://chat.openai.com/`.

[3] OpenAI. *ChatGPT Assistance for Machine Learning Model Loss Functions*. Conversation with ChatGPT (GPT 5.1), November 2025. Available at:`https://chat.openai.com/.`

[4] Center for Strategic and International Studies (CSIS). *Significant Cyber Incidents*. Available at: `https://www.csis.org/programs/strategic-technologies-program/significant-cyber-incidents`.

[5] Cyber Magazine. *The Global Impact of Security Breaches and IT Meltdowns*. Available at: `https://cybermagazine.com/articles/the-global-impact-of-security-breaches-and-it-meltdowns`.

[6] EC-Council. *How to Prevent Network Security Attacks*. Available at: https://www.eccouncil.org/cybersecurity-exchange/network-security/how-to-prevent-network-security-attacks/.

[7] GeeksforGeeks. *Intrusion Detection System (IDS) – Types and Functions*. Available at: https://www.geeksforgeeks.org/ethical-hacking/intrusion-detection-system-ids/.

[8] IBM. *What is a Zero-Day Attack? Understanding Zero-Day Exploits*. Available at: https://www.ibm.com/think/topics/zero-day.

[9] GeeksforGeeks. *Advantages and Disadvantages of Logistic Regression*. Available at: https://www.geeksforgeeks.org/data-science/advantages-and-disadvantages-of-logistic-regression/.

[10] GeeksforGeeks. *Advantages and Disadvantages of Random Forest*. Available at: https://www.geeksforgeeks.org/machine-learning/what-are-the-advantages-and-disadvantages-of-random-forest/.

[11] GeeksforGeeks. *Advantages and Disadvantages of Artificial Neural Networks (ANN) in Data Mining*. Available at: https://www.geeksforgeeks.org/data-analysis/advantages-and-disadvantages-of-ann-in-data-mining/.

[12] GeeksforGeeks. *Advantages and Disadvantages of Logistic Regression*. Available at: https://www.geeksforgeeks.org/data-science/advantages-and-disadvantages-of-logistic-regression/.

[13] GeeksforGeeks. *Advantages and Disadvantages of Random Forest*. Available at: https://www.geeksforgeeks.org/machine-learning/what-are-the-advantages-and-disadvantages-of-random-forest/.

[14] GeeksforGeeks. *Advantages and Disadvantages of Artificial Neural Networks (ANN) in Data Mining*. Available at: https://www.geeksforgeeks.org/data-analysis/advantages-and-disadvantages-of-ann-in-data-mining/.

[15] TensorFlow Keras Documentation. *Introduction to Keras for Engineers*. Available at: https://keras.io/getting_started/.

[16] TensorFlow Keras Documentation. *Binary Crossentropy Loss Function*. Available at: https://www.tensorflow.org/api_docs/python/tf/keras/losses/BinaryCrossentropy.

CODE AVAILABILITY

The complete code for this project is available at: https://github.com/doverara/cmse492_project