Designing a Scalable Machine Learning Pipeline for Cyber Threat Detection

Araynah Dover • CMSE492 • 11/25

Problem & Motivation

Cyber Attacks are increasing and traditional rule-based IDS systems cannot detect new or evolving threats.

Traffic volume is too large for humans to analyze

Detecting anomalies early prevents breaches

Dataset Overview

CICIDS-2017 (all 5 days merged)

2.1 million rows, 105 features

Includes real enterprise traffic + botnet, DoS, brute force, infiltration, XSS, etc.

After cleaning: ≈86 usable features

Target: label_clean → converted to Binary-Label (0 = benign, 1 = attack)

Methodology

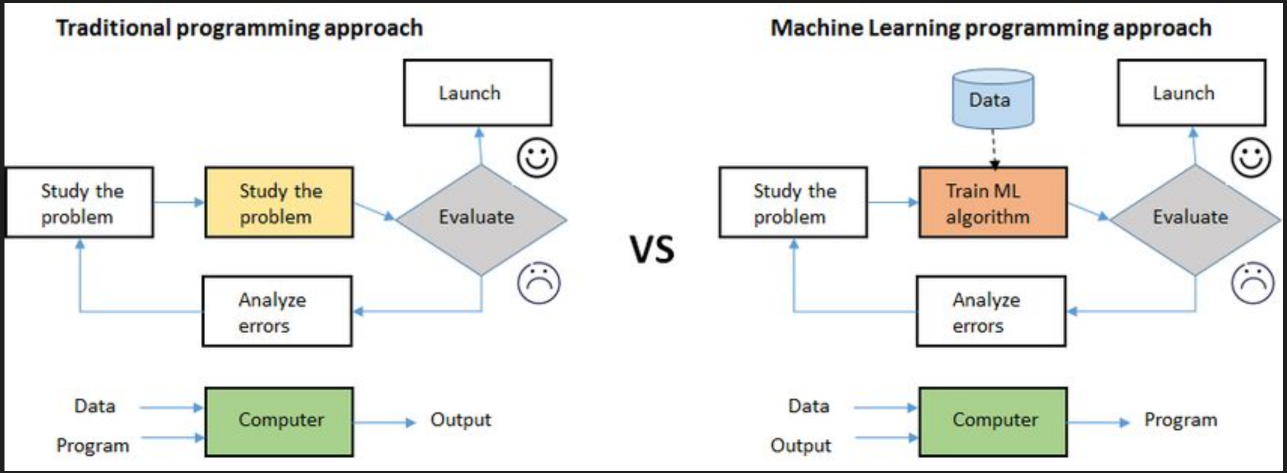
Why ML Works Here

Traditional IDS limitations:

- Only detects known attacks
- Cannot generalize
- High false-positives

Why ML is better:

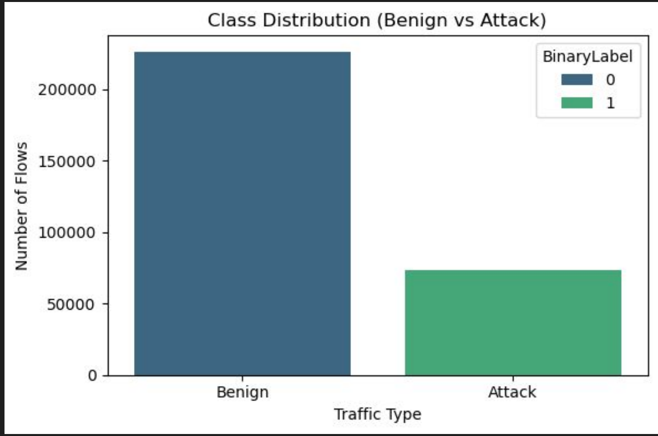
- Learns from statistical patterns
- Detects unseen anomalies
- Scales to millions of flows

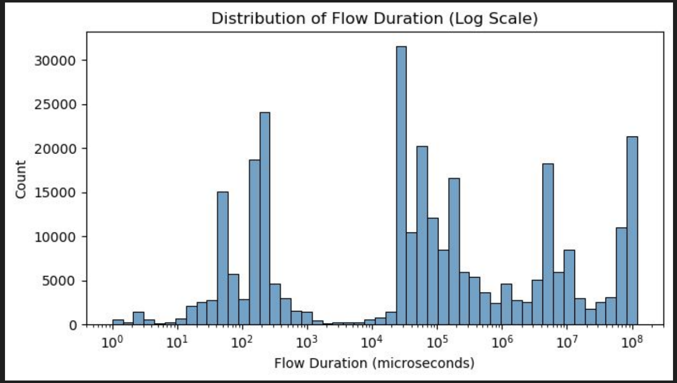


Preprocessing Steps

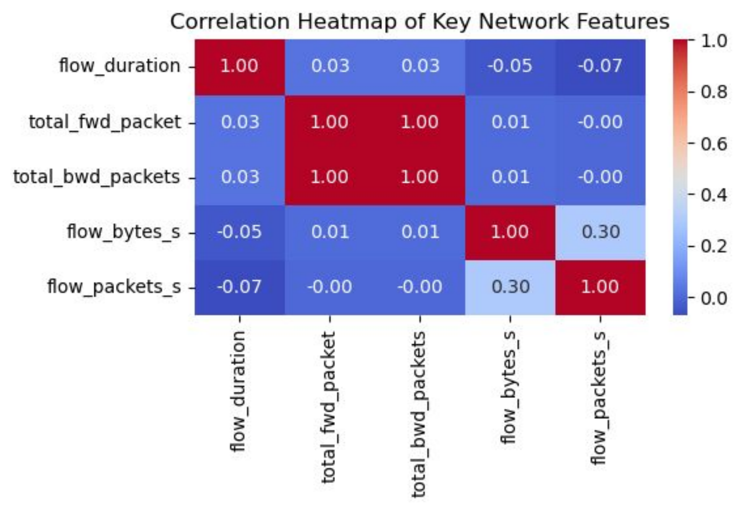
- Dropped irrelevant columns (FlowID, timestamp, local columns)
- Cleaned labels → label_clean → BinaryLabel
- Handle extreme skew and outliers (log scaling on heavy-tailed fields)
- One-hot encoded Protocol
- Train/test split: Stratified 80/20
- No SMOTE → use class_weight='balanced'
- StandardScaler → fit only on training data

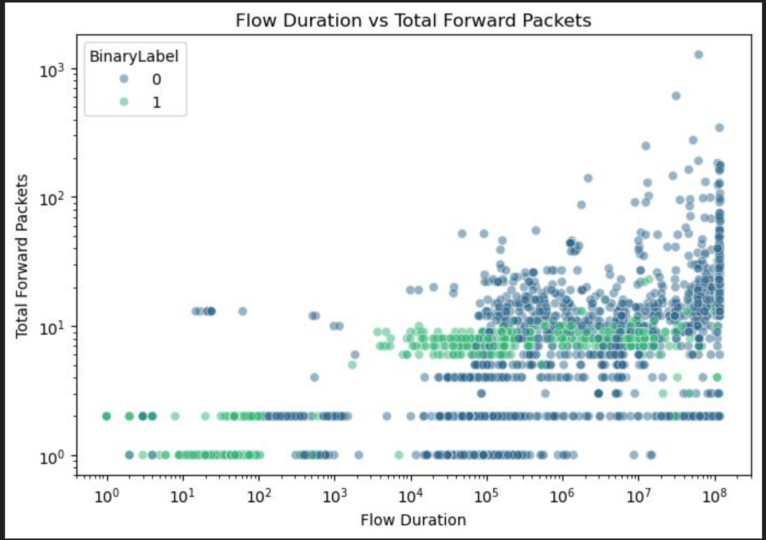
Exploratory Data Analysis





Exploratory Data Analysis





Feature Engineering

- Dropped redundant + low-variance columns
- One-hot encoded Protocol
- Standard scaling

Models Compared

Logistic Regression:

- Linear decision boundary
- Very fast
- Good interpretability
- Uses class_weight='balanced'

Neural Network:

- Can model complex relationships
- Needs regularization
- Harder to interpret

Random Forest:

- Captures nonlinear patterns
- Robust to noise
- Naturally ranks feature importance

Training Strategy

- StandardScaler → only on training data
- class_weight to address imbalance
- 5-fold CV for hyperparameters
- Same train/test split for all models

Evaluation Metrics

- Accuracy
- Precision
- Recall
- F1-score
- ROC-AUC
- Confusion matrices

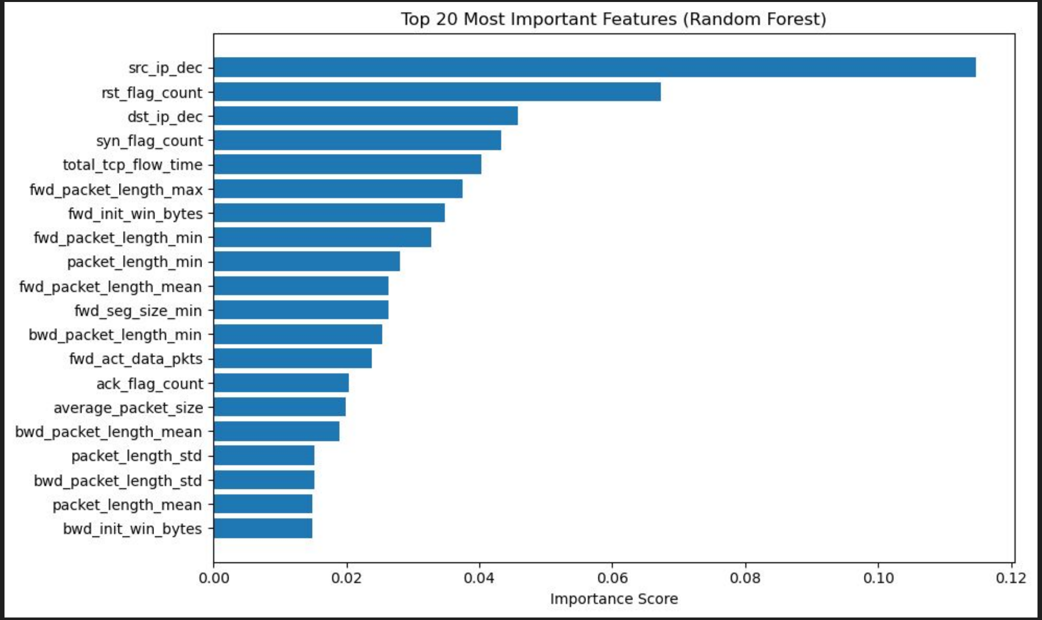
Results

Key Results

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

Model Interpretation

```
git add data/processed/merged.csv
git commit -m "Add merged dataset"
git push origin main
```



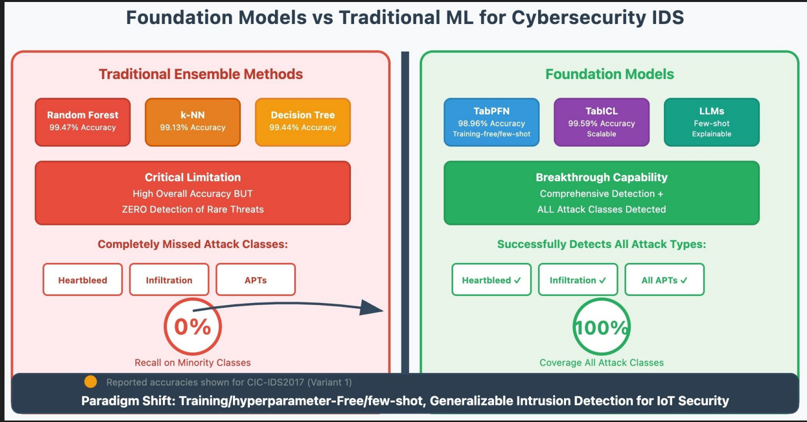
Challenges encountered

SMOTE too large → switched to class weight

SHAP too memory-heavy → removed

Large dataset → had to downsample for EDA

Many useless columns → dropped to speed up pipeline



Future Steps

Try more complex approaches
Try to get a more foundational understanding