# CMSC 25422  Machine Learning for Computer Systems — Final Report

## Introduction

This project looks at the netML Malware Detection dataset. The goal is to classify network flows as either benign or malware using a small set of simple flow features. This type of task matters because modern networks create huge amounts of traffic, and it is not realistic for anyone to manually inspect all of it. Even basic models can help flag suspicious activity early.

My focus was to build a clear pipeline that starts with raw packet captures, extracts features, trains a small group of models, and compares their performance. As was specified in the project instructions, I have formatted my code as a python notebook so that it can easily be run from fron to back by simply clicking “run all.”

## Data and Feature Extraction

The dataset comes from the netML Malware benchmark and contains about 498,000 flows in a compressed pcapng.gz file. Each flow has a metadata field that identifies the type of traffic. I created a binary label by marking any flow whose metadata contained the word “malware” as 1. All other flows were labeled 0.

The dataset is large (about 1.1 GB), so it is not included in the Github repository. There are instructions in the README.md file regarding how to download the dataset and it can be accessed at the following link:

- https://nprint.github.io/benchmarks/malware_detection/netml_malware.html

To run the notebook, the user needs to download the traffic.pcapng.gz file and place it in a data folder.

The label distribution is very imbalanced. There are many more malware flows than benign ones, which affects how the models behave.

For feature extraction, I used simple flow-level properties. These included the number of packets in the flow, the total size in bytes, the mean packet size, the standard deviation of packet sizes, the duration of the flow, and the mean time gap between packets. These features are easy to compute and do not require deep inspection of packet contents.

After processing, I had a feature matrix with shape (498,446 × 6). This gave me enough information to train standard machine learning models.

**Models and Training**

I split the dataset into training and testing sets using an 80 percent to 20 percent split, while keeping the class ratio the same across both sets. Logistic Regression required scaled features, so I standardized those inputs. The tree-based models (Random Forest and HistGradientBoosting) used the raw features.

I trained three models:

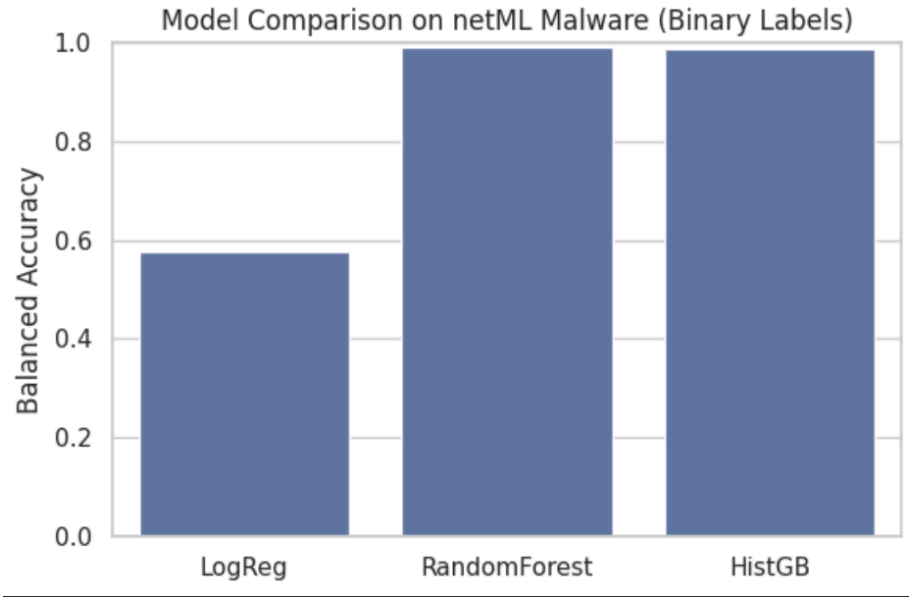
- Logistic Regression with class balancing
- Random Forest with 200 trees
- HistGradientBoostingClassifier with 200 iterations

Balanced accuracy was the main metric I used. This metric is helpful when the classes are imbalanced because it rewards models that do well on both classes instead of only focusing on the majority class.

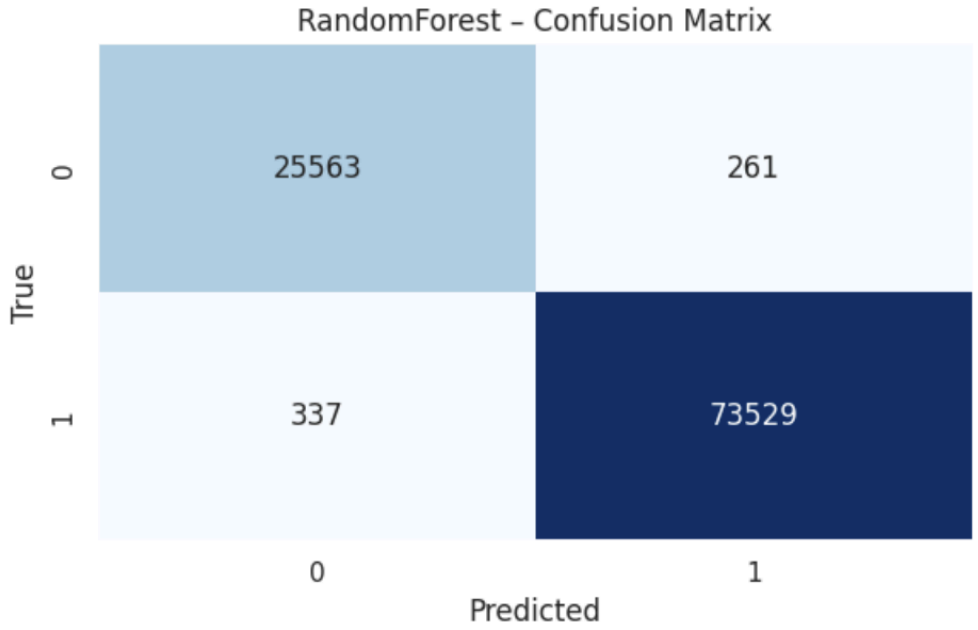
**Results**

Logistic Regression reached a balanced accuracy of about 0.5765. It struggled with both benign and malware flows, especially the benign class.

The Random Forest model performed the best with a balanced accuracy of about 0.9927. It correctly identified almost all the flows with very few mistakes. The HistGradientBoosting model also did very well, reaching about 0.9877. A bar chart comparing all three models shows a clear advantage for the tree-based methods.

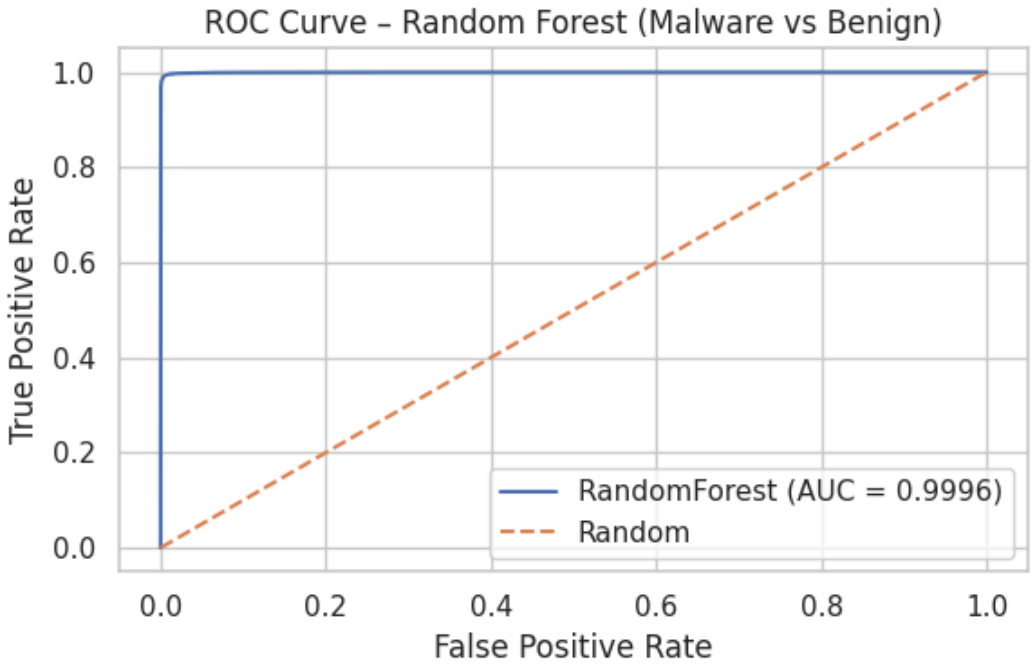


The confusion matrix for the Random Forest model shows that it made very few errors. It misclassified only a small number of benign flows as malware and only a small number of malware flows as benign.

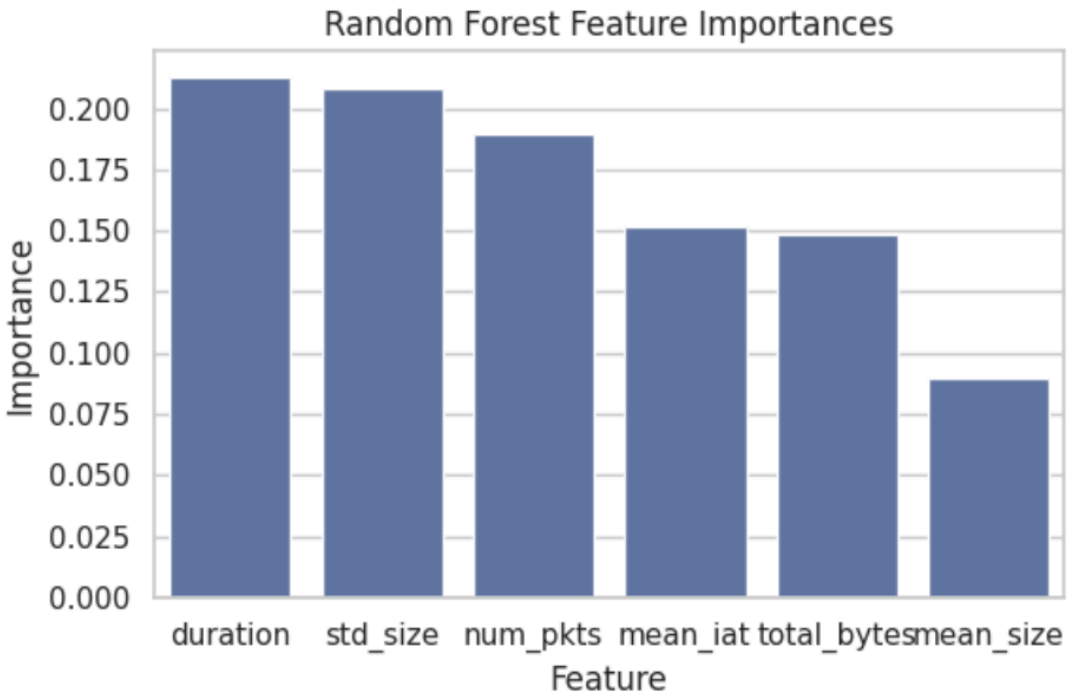



---

I also computed the ROC curve for the Random Forest model. The area under the curve was 0.9996, which means the model is very strong across different classification thresholds.



To understand which features were most useful, I looked at the feature importance values from the Random Forest model. The most important features were the duration of the flow, the standard deviation of packet sizes, and the number of packets. These match common patterns in malware traffic, which often shows unusual timing or size patterns.




---

**Discussion**

This project shows that very simple features combined with standard machine learning models can perform extremely well on the netML Malware Detection dataset. Both tree-based models reached near-perfect performance, while Logistic Regression performed far worse. This suggests that the decision boundary for this problem is not linear.

The work also made it clear how important it is to handle class imbalance properly and to choose evaluation metrics that reflect the actual problem.

If I were to continue this work, I would try using the exact training and testing splits from the official benchmark to compare my results more directly. I would also explore richer features or models that take packet sequences into account.

Overall, this project helped me understand packet-level data processing, feature extraction, training and evaluating machine learning models, and presenting results in a clean and reproducible way.