Cardiff Metropolitan University | Prifysgol Metropolitan Caerdydd

How accurately can supervised machine learning models detect DDoS attacks in the BoT-IoT dataset while keeping false positive rates low enough for use in resource-constrained IoT networks?

Module Leader: Dr Sheikh Tahir Bakhsh

Author: Sid Ali Bendris

Video Link: https://youtu.be/bRVmAoqZG7k

Contents

Figures and Tables

Introduction

The rapid expansion of the Internet of Things (IoT) has led to the deployment of billions of interconnected devices in homes, industry, healthcare, transport and critical infrastructure. These devices continuously collect, process and exchange data, often operating with minimal human intervention. While this pervasive connectivity enables automation and new digital services, it also significantly enlarges the potential attack surface of modern networks, this is due to the fact that many IoT devices are deployed with weak default configurations, limited built-in security mechanisms and poor patch management, making them attractive targets for attackers. Among the various threats that affect IoT environments, Distributed Denial-of-Service (DDoS) attacks are particularly disruptive. In a DDoS attack, a large number of compromised devices are coordinated to send excessive traffic towards a victim service or network, exhausting its bandwidth, processing capacity or memory, and preventing legitimate users from accessing resources. High-profile incidents have shown how compromised IoT devices can be organised into botnets capable of generating massive amounts of malicious traffic. Because IoT deployments often support critical services, the loss of availability caused by DDoS attacks can have significant operational and economic impact. To defend against such attacks, many organisations rely on Intrusion Detection Systems (IDS) that monitor network traffic and raise alerts when suspicious patterns are detected. Traditional signature-based IDS struggle to detect novel or evolving attack behaviours, particularly in heterogeneous IoT environments where traffic characteristics can vary widely. Machine learning-based IDS have therefore attracted considerable interest, as they can be trained to distinguish benign from malicious traffic by learning statistical patterns from labelled datasets. Publicly available datasets such as BoT-IoT provide examples of both normal and attack traffic, including DDoS scenarios, and are widely used to train and evaluate detection models.

However, deploying machine learning-based IDS in IoT networks presents several practical challenges. Many IoT devices and edge gateways operate under strict resource constraints, with limited processing power, memory and energy budgets. Complex models may be too costly to run in real time on such platforms, especially if they require frequent retraining or high-dimensional feature sets. In addition, a model that achieves high overall accuracy can still be impractical if it produces a significant number of false positives. In resource-constrained environments, false positives can lead to legitimate traffic being blocked, unnecessary consumption of limited resources, and reduced trust in the detection system by administrators. These considerations highlight the need to understand not only how accurately machine learning models can detect DDoS attacks, but also how well they control false positive rates in realistic IoT settings. This study therefore focuses on supervised learning approaches applied to the BoT-IoT dataset, with an emphasis on the trade-off between detection performance and false positives. Specifically, it investigates the following research question: **How accurately can supervised machine learning models detect DDoS attacks in the BoT-IoT dataset while keeping false positive rates low enough for use in resource-constrained IoT networks?**

To address this question, the report evaluates three supervised classification models—logistic regression, decision tree and random forest—on a binary classification task that distinguishes benign traffic from DDoS traffic within the BoT-IoT dataset. The models are trained and tested using a reproducible experimental pipeline that includes data preprocessing, feature selection and performance evaluation using a range of metrics, including accuracy, precision, recall, F1-score and false positive rate. The remainder of the report is organised as follows. Section 2 reviews related work on machine learning-based intrusion detection for IoT and DDoS attacks. Section 3 describes the BoT-IoT dataset and the experimental methodology used in this study. Section 4 presents and analyses the results of the experiments. Section 5 discusses the implications of the findings, outlines best practices for deploying IDS in resource-constrained IoT environments, and reflects on the limitations of the work, before Section 6 concludes the report.

Literature Review

I have chosen 3 recent research papers that are closely related to my subject topic to analyse and dissect the different methodology used, find gaps in their research or anyways the research could have been improved and finally discuss the results of each paper.

Botnet-Based DDoS Detection in IoT Using BoT-IoT (Pokhrel et al.)

Pokhrel et al. propose a machine-learning based model to detect botnet-driven DDoS attacks in IoT networks, using the BoT-IoT dataset as the main source of training and evaluation data. Their motivation is that most prior work on botnet detection either relies on non-IoT datasets or ignores the effects of class imbalance on model performance. The authors explicitly focus on botnet DDoS as one of the most critical threats to IoT environments. The study uses a single BoT-IoT CSV file containing approximately 1 million records, of which over 99% are labelled as botnet traffic and less than 1% as normal, creating a highly imbalanced dataset. The authors first perform data cleaning (dropping rows with null values), then normalise numerical features with min–max scaling and transform categorical protocol and state fields into numeric form. Feature engineering is carried out using chi-square scores: only the eight features with an F-score above the mean (bytes, sbytes, dbytes, rate, pkts, spkts, srate and drate) are retained to reduce dimensionality and computation.To address the extreme class imbalance, Pokhrel et al. generate a second dataset by applying the Synthetic Minority Oversampling Technique (SMOTE) to oversample the minority (normal) class, producing a balanced dataset with equal numbers of botnet and normal records. Three supervised classifiers are evaluated: Gaussian Naïve Bayes, K-Nearest Neighbours (KNN), and a multilayer perceptron artificial neural network (MLP-ANN). The experiments are run on both the original imbalanced dataset and the SMOTE-balanced version, with 80/20 train–test splits and 5-fold cross-validation. Performance is assessed using confusion matrices, accuracy, precision, recall, F1-score and ROC-AUC.Results show that accuracy alone is misleading on the imbalanced dataset: Gaussian Naïve Bayes achieves ~99% accuracy but only around 60% ROC-AUC and very poor recall and F1-score for the minority class. After balancing with SMOTE and applying feature selection, KNN emerges as the best performing classifier, achieving about 92% accuracy and 92.2% ROC-AUC on the balanced dataset and over 99% accuracy and 99.2% ROC-AUC on the imbalanced data. The authors conclude that KNN, combined with SMOTE and feature engineering, is an effective approach for botnet DDoS detection in BoT-IoT. However, the study focuses primarily on classifier choice and the impact of class imbalance; it does not consider resource constraints of IoT deployments, and false positive rates are only indirectly discussed via the confusion matrices rather than as a design objective.

Multi-Level ML-Based IDS Using BoT-IoT (Alosaimi & Almutairi)

Alosaimi and Almutairi present an intrusion detection system for IoT networks that uses BoT-IoT to train a three-level machine-learning pipeline. Their goal is to detect a wide range of IoT attacks—particularly DoS/DDoS—while addressing dataset imbalance and achieving very high detection accuracy. The three levels correspond to: (i) binary classification (attack vs normal), (ii) four attack categories (DDoS, DoS, Reconnaissance, Theft), and (iii) ten attack subcategories (e.g. DDoS_HTTP, DoS_TCP, Reconnaissance OS fingerprinting, keylogging).From the original BoT-IoT dataset, the authors derive three working databases. The first is a large multi-attack dataset (DB1). The second (DB2) is produced by random sampling to reduce the number of instances from 3.6 million to around 63,000 records. The third (DB3) addresses class imbalance by applying SMOTE to create a more balanced version with approximately 90,600 records. All three undergo preprocessing steps including handling missing values, min–max normalisation and encoding of categorical features such

as protocol, category and subcategory. Five supervised algorithms are trained and evaluated at each level: decision trees, an ensemble bagging classifier, KNN, linear discriminant analysis (LD) and support vector machines (SVM). The dataset is split 70/30 into training and testing sets. Evaluation uses accuracy, error rate, recall, specificity, precision, F-measure and training/testing time. Tables 2–7 (pages 11–12) show that for both DB2 and DB3 the ensemble bagging classifier and decision trees consistently achieve near-perfect performance, often reporting 100% accuracy, recall and specificity across all three levels. The paper demonstrates that relatively standard ML models, when combined with sampling and preprocessing, can achieve extremely high accuracy on BoT-IoT for multi-class intrusion detection, not just binary attack detection. It also recognises the issue of class imbalance and explicitly addresses it via SMOTE. However, such uniformly high metrics raise questions about overfitting and generalisability beyond the BoT-IoT environment. The focus is on overall accuracy rather than on operational measures such as false positive rates, and the work does not explicitly explore trade-offs between model complexity, detection quality and suitability for resource-constrained IoT devices.

Real-Time INIDS with Multiple Classifiers on BoT-IoT (Ashraf et al.)

Ashraf et al. develop INIDS, a real-time intrusion detection system for IoT networks, also trained on the BoT–IoT dataset. Their primary contribution is a systematic comparison of multiple machine-learning classifiers on a large, balanced subset of BoT–IoT, with the aim of determining which algorithms are most suitable for IoT-based NIDS. The authors select approximately 3.6 million records from BoT–IoT and construct a balanced dataset using a hybrid sampling strategy that combines random oversampling, SMOTE and random under sampling. Minority classes such as keylogging and data theft are oversampled, while majority classes are reduced to avoid dominance. They then perform extensive preprocessing: cleaning missing or erroneous values, transforming categorical labels, and applying feature selection driven by the Pearson correlation coefficient. Starting from 26 engineered features, they derive three feature sets (14, 11 and 10 features) by progressively removing highly correlated attributes. A random forest feature-importance analysis (Figure 8, page 9) is used to verify that the selected features—such as subcategory_UDP, stddev, drate, state_number and proto_tcp—are the most influential for classification.Seven supervised classifiers are trained: logistic regression, SVM, KNN, decision tree, random forest, Naïve Bayes and a feed-forward artificial neural network. Each model is trained three times, once for each feature set, using an 80/20 train–test split on a binary task (attack vs benign). The authors report accuracy for each feature set (Table 3, page 10), then analyse robustness to feature reduction and potential over/underfitting. With the 14-feature set, random forest achieves the best accuracy (99.2%), followed by Naïve Bayes (98.8%) and KNN (97.1%), while simpler models such as logistic regression perform notably worse.Beyond accuracy, the authors also examine precision, recall, F1-score, specificity and AUC-ROC (Table 5 and Figures 11–15). Random forest again provides the strongest performance, with recall 0.993, F1-score 0.997 and AUC 0.93, while Naïve Bayes achieves recall 0.989 and F1-score 0.994. They additionally discuss computational cost and scalability (Table 4), noting that although random forest is more expensive to train than logistic regression or Naïve Bayes, its training and testing times remain manageable on modern hardware and can be parallelised. Nonetheless, the evaluation is still framed in terms of accuracy and aggregate metrics rather than explicit false positive constraints, and the experiments focus on generic attack detection rather than specific DDoS detection for IoT.

Critical Discussion and Research Gap

Across these three studies, several common patterns emerge. All of them adopt BoT-IoT as a realistic IoT attack dataset and use supervised machine-learning classifiers to build intrusion detection solutions. They also recognise that raw BoT-IoT is highly imbalanced and apply some form of resampling—SMOTE in Pokhrel et al. and Alosaimi & Almutairi, hybrid oversampling and

undersampling in Ashraf et al.—combined with feature engineering to improve performance. However, their problem focus and modelling choices differ. Pokhrel et al. restrict their scope to botnet-based DDoS detection, using a relatively simple three-model comparison (Gaussian NB, KNN, MLP-ANN) and a single BoT-IoT CSV file. Their main insight is that balancing and feature selection are essential to avoid illusory accuracy on highly imbalanced data, with KNN emerging as the most effective classifier in their setting. In contrast, Alosaimi & Almutairi and Ashraf et al. both address multi-attack intrusion detection, but at different levels of granularity: the former uses a three-level label hierarchy spanning ten attack subtypes, while the latter focuses on binary attack/benign classification across many attack families.There are also differences in algorithm breadth and evaluation depth. Pokhrel et al. evaluate only three classifiers but analyse imbalanced vs balanced performance in some detail using ROC-AUC and cross-validation. Alosaimi & Almutairi test five models and report multiple metrics, yet many of their results are effectively perfect (100% accuracy, recall and specificity) on both subsampled and SMOTE-balanced data, which raises concerns about overfitting to BoT-IoT. Ashraf et al. provide the broadest comparison, training seven classifiers on millions of records and examining robustness across different feature sets, as well as including an explicit scalability discussion with approximate training and inference times.

Despite this, three important gaps remain in the context of the present project:

1. **DDoS-specific detection vs generic intrusion detection.**
   Only Pokhrel et al. focus explicitly on botnet-based DDoS attacks in IoT, but their experiments are limited to KNN, NB and MLP and use a relatively small subset of BoT-IoT.

- The more recent works treat DDoS as just one of many attack types and do not provide a targeted analysis of DDoS detection performance, making it difficult to understand how well their models would perform if deployed specifically to defend IoT networks against volumetric attacks.
- **False positives and IoT resource constraints are under-explored.**
  All three papers mainly emphasise overall accuracy and related aggregate metrics. While Pokhrel et al. and Ashraf et al. report ROC-AUC and confusion-matrix-derived measures, none of the studies explicitly design or evaluate models around **low false positive rates** as a primary constraint, which is critical in resource-constrained IoT environments where unnecessary blocking or alerting can be costly. Similarly, only Ashraf et al. discuss computational cost in any detail, and even then at the level of desktop-class hardware rather than constrained IoT gateways.
- **Limited analysis of simple, interpretable models for edge deployment.**
  Random forest, ensemble bagging and deep neural networks consistently appear as the strongest performers in terms of raw accuracy. However, these models can be relatively heavy for deployment on IoT gateways or embedded devices. Logistic regression and decision trees are included in Ashraf et al., but they are not studied in a DDoS-specific context and are not evaluated primarily for their trade-off between simplicity, resource usage and acceptable false positive rates.

Taken together, the existing literature shows that BoT-IoT is a suitable benchmark for ML-based intrusion detection in IoT environments and that classic supervised learning methods can achieve very high detection accuracy. At the same time, there is still a need for a focused study that: narrows the problem to DDoS detection using BoT-IoT, compares lightweight supervised models such as logistic regression, decision trees and random forest, and evaluates them explicitly in terms of detection performance and false positive rates, with an eye towards deployment in resource-constrained IoT networks.

Comparative table

Study	Dataset / Scenario	Attack types	Mitigation technique	Effectiveness	Deployment feasibility
Pokhrel et al., 2021 – "IoT Security: Botnet detection in IoT using Machine Learning"	BoT-IoT (single CSV file, ~1M records; highly imbalanced: >99% botnet, <1% normal)	Botnet-based DDoS attacks in IoT networks (binary: botnet vs normal traffic)	ML-based network IDS using three supervised models: Gaussian Naïve Bayes, K-Nearest Neighbours (KNN) and MLP-ANN. Pipeline includes data cleaning, min–max normalisation, chi-square–based feature selection (top 8 traffic/flow features), and SMOTE oversampling to balance the minority normal class.	On the imbalanced dataset, Naïve Bayes shows very high accuracy but poor recall and ROC-AUC for the minority class, demonstrating misleading performance. After SMOTE and feature selection, KNN is the most stable: it achieves high accuracy and ROC-AUC on both balanced and imbalanced data, with substantially improved recall/F1 for normal traffic.	Focus is mainly on algorithm selection and imbalance handling, not on deployment. No explicit resource or latency analysis for IoT devices. KNN is argued to be "effective" for botnet detection, but feasibility on constrained IoT gateways or devices (memory/CPU cost, online processing) is not evaluated, and false positives are not treated as a primary design constraint.
Alosaimi & Almutairi, 2023 – "An Intrusion Detection System Using BoT-IoT"	BoT-IoT transformed into three databases: full multi-attack set (DB1), randomly down-sampled subset (DB2), and SMOTE-balanced subset (DB3).	Multiple IoT attack families across three levels: (1) attack vs normal; (2) four categories – DDoS, DoS, information gathering (scanning, OS fingerprinting), information theft (keylogging, data theft); (3) ten attack sub-types (e.g. DDoS_TCP/UDP/HTTP).	Multi-level ML-based IDS: five classifiers (Decision Tree, Bagging ensemble, KNN, Linear Discriminant, SVM) trained at each level. Extensive preprocessing: missing-value handling, normalisation, categorical encoding, random sampling and SMOTE for class balance. The system is framed as an AI-based IDS that monitors network signals and classifies them into attack types.	For both sampled and SMOTE-balanced datasets, Decision Tree and Bagging ensemble consistently achieve near-perfect performance (often ≈100% accuracy, recall and specificity) across binary, category and sub-type levels. This indicates very strong separation on BoT-IoT but also raises concerns about overfitting and realism beyond the testbed.	Authors present a conceptual IDS architecture (sensors + recorder + AI-based IDS) and suggest the approach could secure IoT networks by isolating devices or blocking malicious traffic. However, they do not quantify resource usage, latency or memory footprint, nor do they discuss where exactly in an IoT stack (device, gateway, fog, cloud) each model would run. False positive rates are reported via standard metrics but are not explicitly optimised or constrained, so deployment feasibility in resource-constrained IoT environments remains largely theoretical.

Ashraf et al., 2025 – “Making a Real-Time IoT Network Intrusion-Detection System (INIDS) Using a Realistic BoT–IoT Dataset”	Large BoT-IoT subset (~3.6M records) with hybrid sampling (random oversampling, SMOTE, and undersampling) to build a balanced dataset spanning multiple attack families. Features reduced from 26 to 14/11/10 via Pearson correlation and Random Forest feature importance	Binary attack vs benign detection over a mix of BoT-IoT attacks (including DDoS, DoS, reconnaissance and information-theft traffic). Not DDoS-only, but DDoS is one of the main attack categories present.	“INIDS” – a real-time NIDS using seven supervised classifiers: Logistic Regression, SVM, KNN, Decision Tree, Random Forest, Naïve Bayes and ANN. Workflow: extensive cleaning, encoding, hybrid balancing, correlation-based feature selection and iterative training on three feature sets to study robustness under feature reduction.	With the 14-feature set, Random Forest achieves the best results (≈99.2% accuracy, very high precision/recall/F1), followed by Naïve Bayes and KNN; simpler models (e.g. Logistic Regression) underperform. The system maintains strong performance even when features are reduced, indicating a robust feature subset for BoT-IoT-based intrusion detection.	INIDS explicitly considers runtime performance and deployment: the authors discuss training and testing times and describe the model as “lightweight” after optimisation. They propose deployment at three layers: (i) on resource-constrained IoT devices where feasible, (ii) at the edge/fog layer for local traffic analysis, and (iii) with cloud integration for central monitoring and scalability. However, while they mention lightweight design and possible placements, they still focus primarily on accuracy and general metrics, rather than on strict false-positive constraints or detailed hardware-level resource profiling for IoT gateways.

Table 1 Research Comparison Table

Implementation Study/Case Analysis

Dataset description – BoT-IoT

This project uses the BoT-IoT dataset, which is generated from a simulated IoT network consisting of legitimate devices and compromised hosts producing different attack types, including DDoS. From the full collection of CSV files, a subset was constructed named "data_t.csv containing only records labelled DDoS or Normal in the category field, so that the problem becomes a binary classification task: detect DDoS vs normal traffic.

For the main experiment reported here, a single combined CSV (data_t.csv) was used, containing **1,109 flows** in total: **1,044 DDoS** and **65 Normal**. A stratified 70/30 split produced a training set with **731 DDoS** and **45 Normal** flows, and a test set with **313 DDoS** and **20 Normal** flows.

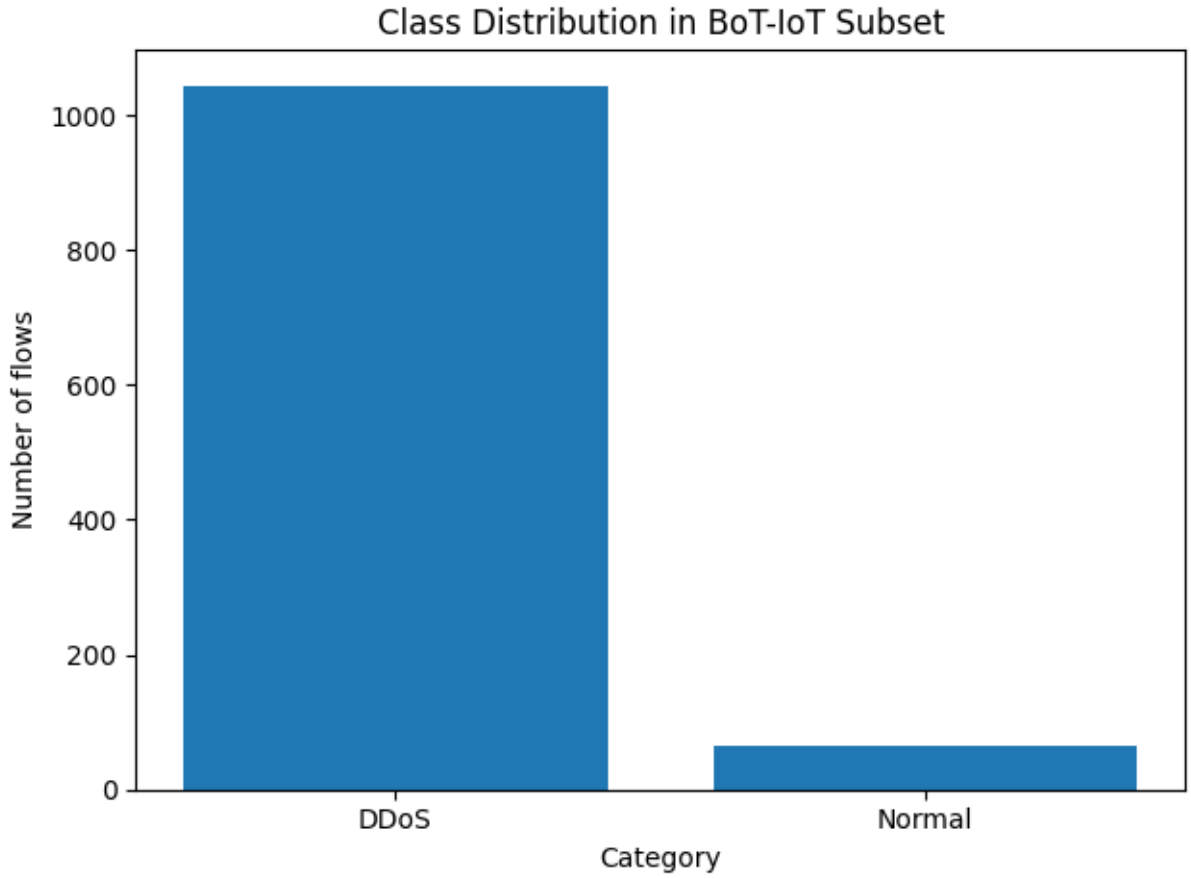

Figure 1 Class Distribution in BoT-IoT Subset

Set	**Total flows**	**DDoS**	**Normal**
Train	776	731	45
Test	333	313	20
Overall	1,109	1,044	65

Table 2 Dataset Composition

Figure 3.1 and Table 3.1 highlight that even in this “more balanced” subset, DDoS traffic still dominates normal traffic, which has implications for evaluating false positive rates later.

Feature-wise, each row represents a network flow summarised by:

- basic flow statistics (pkts, bytes, spkts, dpkts, sbytes, dbytes),
- timing information (stime, ltime, dur, mean, stddev, min, max),
- rate features (rate, srate, drate),
- protocol and TCP flag information (proto, flgs, state),
- and addressing fields (saddr, daddr).

These features are typical of flow-based IDS designs and capture both volume and behavioural characteristics of the traffic.

Tools and environment

All experiments were implemented in Python 3 using Google Colab. The main libraries were:

- **pandas** and **numpy** for data handling and transformation,
- **scikit-learn** for preprocessing, model training, and evaluation,
- **matplotlib** for generating plots and visualisations.

The notebook was executed on Colab’s standard CPU runtime, which is comparable to a modest laptop CPU. This is useful when reasoning about the feasibility of deploying similar models on resource-constrained IoT gateways.

Preprocessing pipeline

The preprocessing pipeline mirrors what would be needed to deploy a flow-based IDS on live traffic and was kept consistent across all models.

1. **Label construction**

The BoT-IoT category field was used as the only source of class labels.

- category = "DDoS" was mapped to label **0**.
- category = "Normal" was mapped to label **1**.

LabelEncoder was used to perform this mapping; inspecting the resulting classes confirmed the mapping 0 → DDoS, 1 → Normal. Throughout the analysis, DDoS is treated as the positive class and Normal as the negative class.

2. **Removal of label fields**

To avoid data leakage, the following columns were removed from the feature set and used only for labelling and filtering:

- attack (binary 0/1 attack flag),
- category (string label),
- subcategory (fine-grained attack type).

3. **One-hot encoding of categorical features**

Several features are categorical strings (e.g. tcp, udp, combinations of TCP flags). These were one-hot encoded into Boolean indicator columns using pandas.get_dummies:

- proto – transport protocol,
- flgs – TCP flag combinations,
- state – connection state,
- saddr and daddr – source and destination IP addresses.

After encoding, the original string columns were dropped, leaving only numeric and Boolean (0/1) features. Although one-hot encoding IP addresses risks overfitting to specific hosts, it reflects typical testbed practice and is acknowledged as a limitation in the Discussion.

4. **Handling missing values**

Some columns in the BoT-IoT subset contained only missing values (e.g. MAC/OUI-related fields). These all-NaN columns were removed entirely. Remaining missing values were imputed using median imputation (SimpleImputer(strategy="median")), which is robust to outliers and preserves the general distribution of each feature.

5. **Feature scaling**

All remaining features were standardised to zero mean and unit variance using StandardScaler. This step is essential for Logistic Regression and generally beneficial for distance-based or gradient-based models, ensuring that high-magnitude features (e.g. byte counts) do not dominate the learning process.

6. **Train–test split**

The final processed dataset was split into training and testing sets using a 70/30 stratified split (train_test_split(..., stratify=y)), preserving the DDoS/Normal ratio in both sets. All three models were trained on the same training data and evaluated on the same held-out test set to allow fair comparison.

Models

Three supervised classifiers were selected, representing a spectrum from simple linear to ensemble methods, but all widely used in IDS research:

1. **Logistic Regression**

Logistic Regression provides a linear decision boundary and is computationally lightweight, making it a strong candidate for deployment on IoT gateways. The scikit-learn implementation was used with max_iter increased to 200 to ensure convergence after feature scaling. No regularisation tuning was performed; the default L2 regularisation was retained.

2. **Decision Tree**

The Decision Tree classifier can learn non-linear rules directly from the data and offers some interpretability in terms of "if–then" path conditions. The default Gini impurity criterion was used. Tree depth and other hyperparameters were left at their defaults to keep the setup simple and to focus on high-level behaviour rather than fine-grained optimisation.

3. **Random Forest**

Random Forest is an ensemble of decision trees trained on bootstrap samples with feature subsampling, typically offering stronger generalisation at the cost of higher computational overhead. The default configuration (n_estimators=100) was adopted. Random Forest also provides feature importance scores, which were later used to understand which flows features dominated the decision process.

Each model was trained on X_train, y_train and evaluated on X_test, y_test. Metrics computed for DDoS (positive class) included accuracy, precision, recall, F1-score, and false positive rate (FPR), where FPR is defined as:

$$\text{FPR} = \frac{\text{FP (Normal misclassified as DDoS)}}{\text{FP + TN (all actual Normal flows)}}$$

Results

The aggregate metrics for the three models on the test set (313 DDoS, 20 Normal) are shown below.

Model	**Accuracy**	**Precision (DDoS)**	**Recall (DDoS)**	**F1 (DDoS)**	**FPR (Normal)**	**TP (DDoS)**	**FN (DDoS)**	**FP (Normal)**	**TN (Normal)**
Logistic Regression	1.000	1.000	1.000	1.000	0.000	313	0	0	20
Decision Tree	1.000	1.000	1.000	1.000	0.000	313	0	0	20
Random Forest	1.000	1.000	1.000	1.000	0.000	313	0	0	20

Table 3 Model Performance on test set

All three models achieved perfect scores on this subset: every DDoS flow in the test set was correctly detected, and every Normal flow was correctly identified as benign. This yields FPR = 0 for Normal traffic in all cases.

The confusion matrices provide a clearer visual representation of these results.

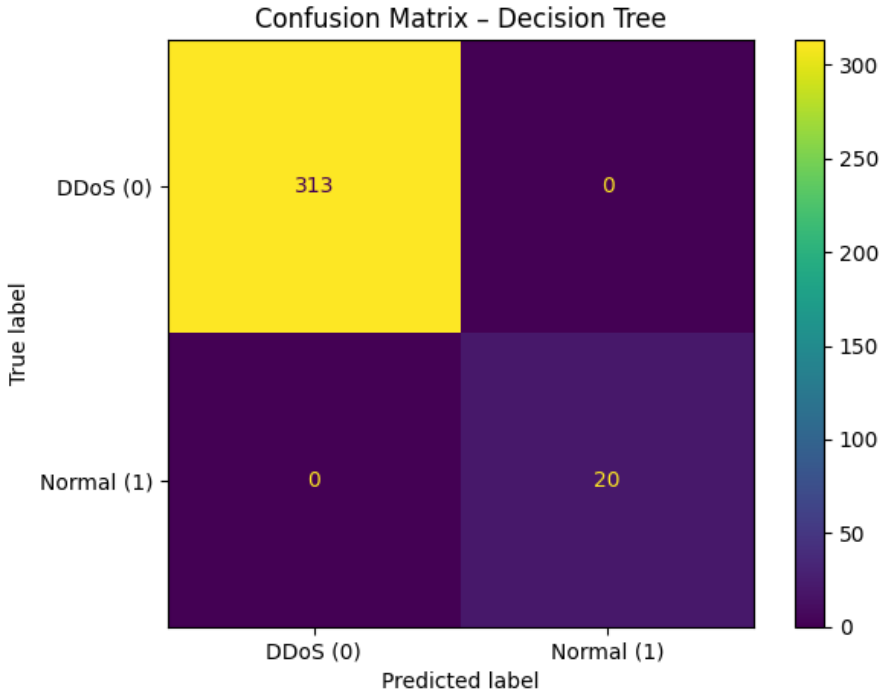

Figure 2 Confusion Matrix (Decision Tree)

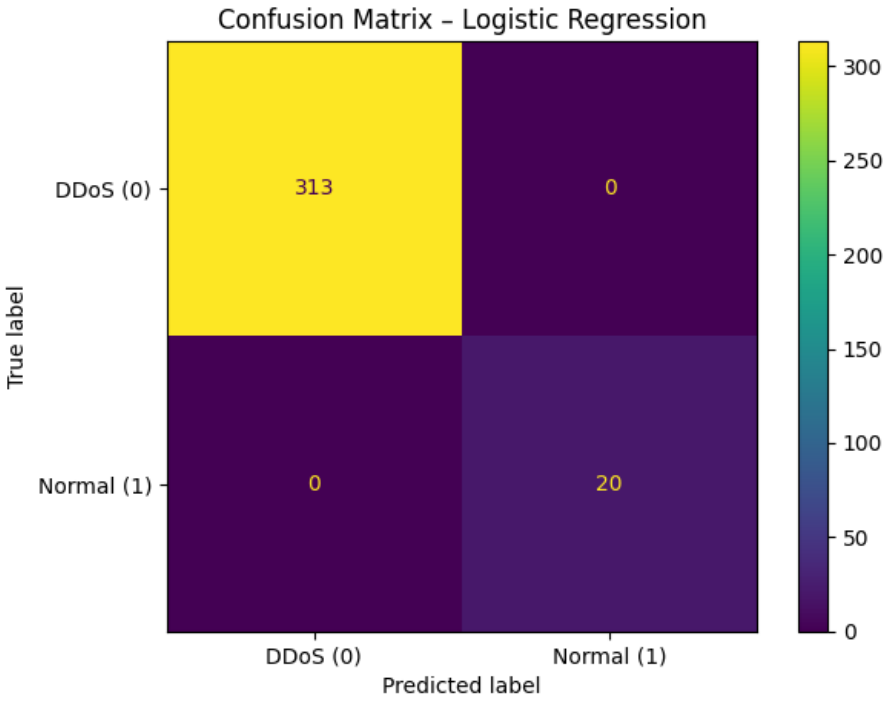

Figure 3 Confusion Matrix (Logistic Regression)

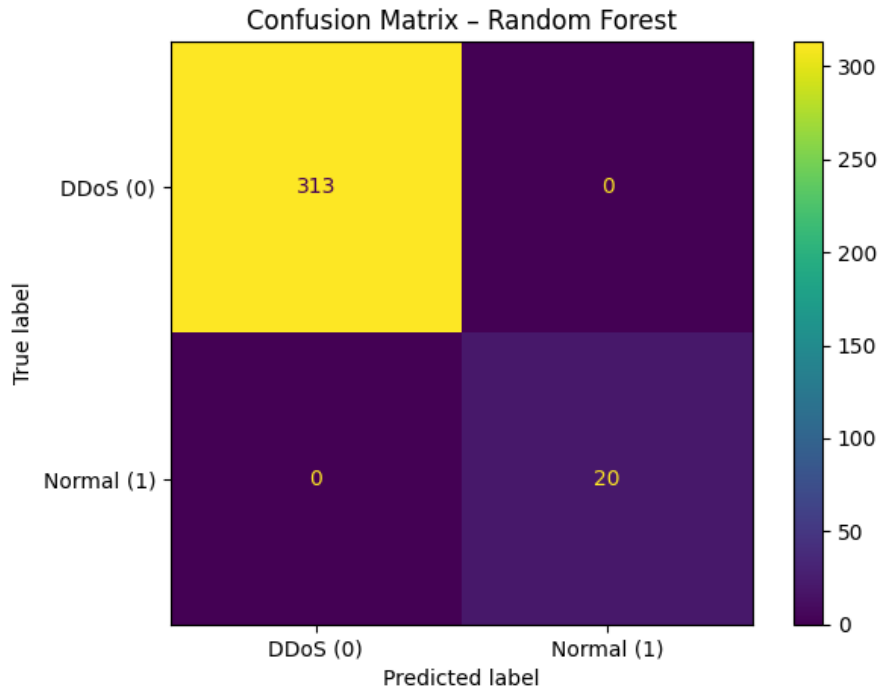

Figure 4 Confusion Matrix (Random Forest)

Because the confusion matrices are identical, only one would be strictly necessary in the report; however, including two (e.g. Logistic Regression and Random Forest) allows you to emphasise that both simple and ensemble models behave identically on this subset.

To compare the models numerically, a bar chart of recall and F1-score for DDoS was created.

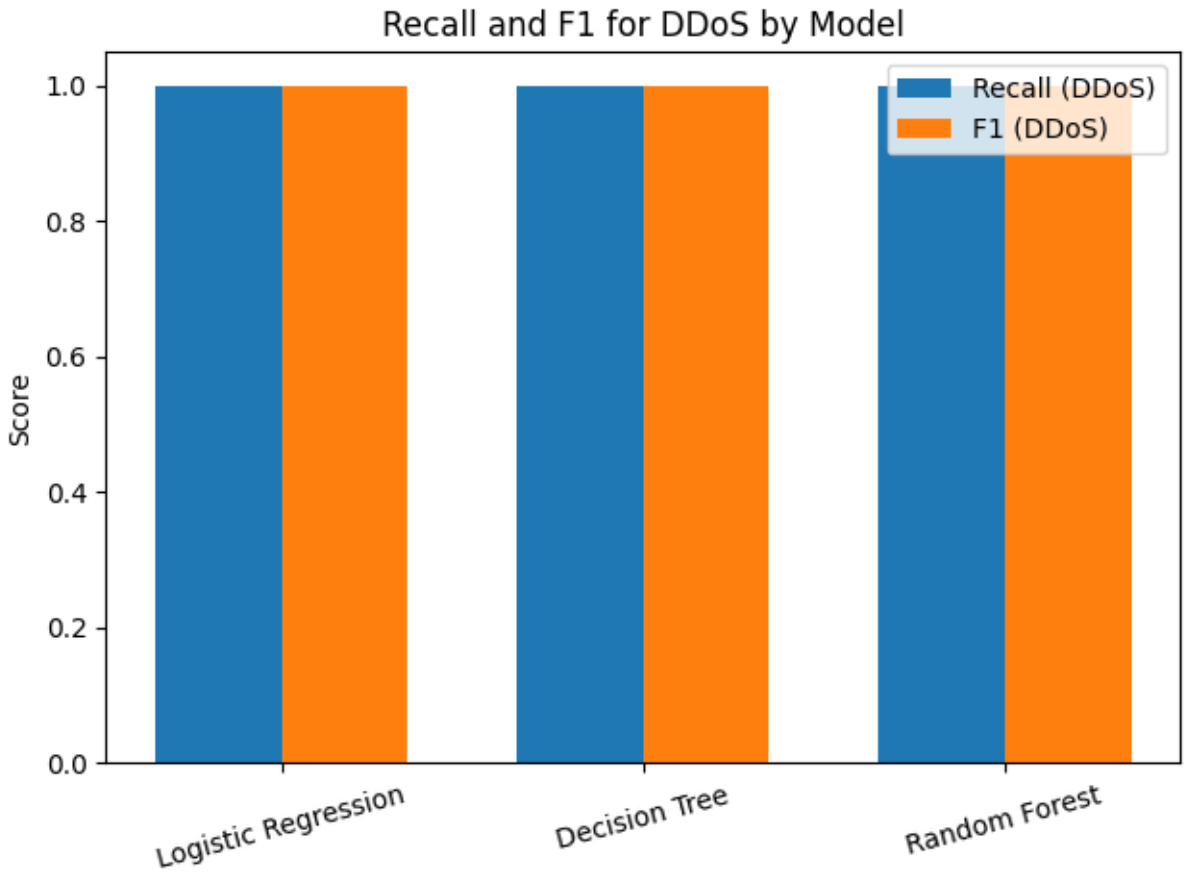

Figure 5 Recall and F1 for DDoS by Model

Unsurprisingly, recall and F1 are equal to 1.0 for all three models. A separate plot of FPR on Normal traffic shows all bars at zero.

Figure 6 False Positive Rate on Normal Traffic

Finally, ROC curves for Logistic Regression and Random Forest were generated using predicted probabilities.

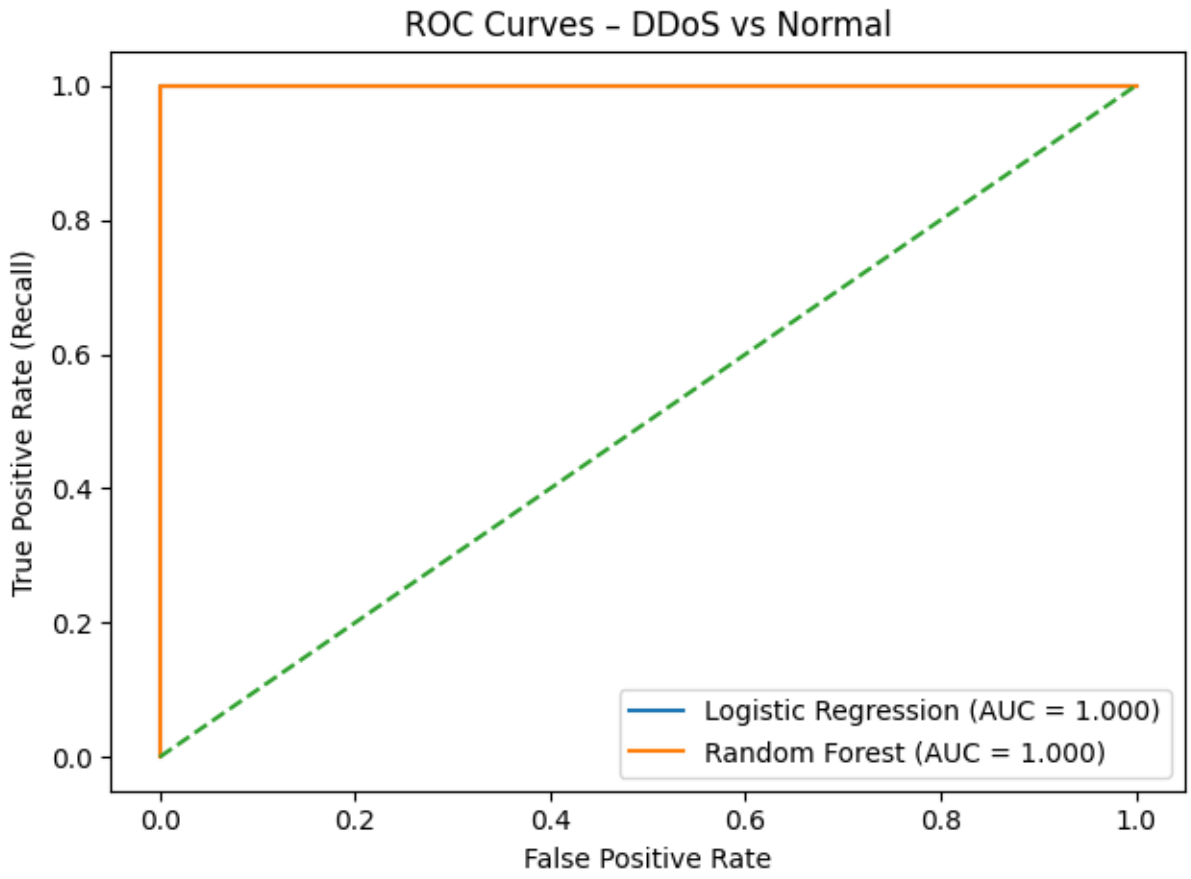

Figure 7 ROC Curves (DDoS vs Normal)

Both curves hug the top-left corner with **AUC = 1.0**, further confirming that the DDoS and Normal classes are linearly separable in this subset.

Interpretation

The results indicate that, for this BoT-IoT subset, DDoS and Normal flows are extremely separable using relatively straightforward supervised learning. Even the simplest model, Logistic Regression, achieves perfect detection with zero false positives, which is highly desirable in IoT environments where benign traffic should not be blocked.

However, the perfect metrics must be interpreted in light of the dataset characteristics:

- The subset is still dominated by DDoS traffic and contains only 65 Normal examples overall (20 in the test set). This means that FPR is estimated from a very small number of benign samples; a single misclassification would have increased FPR by 0.05.
- One-hot encoding of IP addresses (saddr_*, daddr_*) allows models to memorise attacker/victim hosts specific to the BoT-IoT testbed. This likely contributes to the perfect separation and may not generalise to other networks where IPs and device populations differ.

Random Forest feature importance scores were examined to understand which features drive decisions.

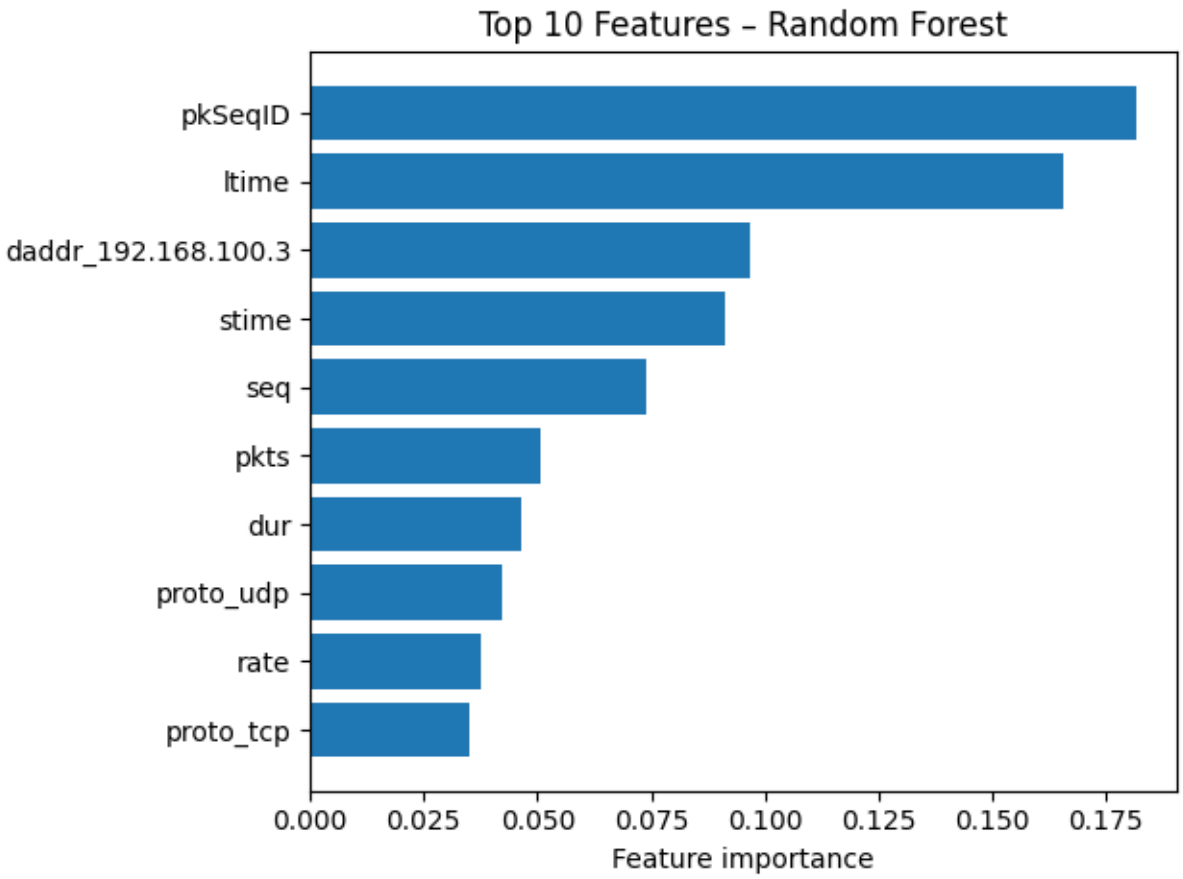

Figure 8 Top 10 Features (Random Forest)

The most influential features include pkSeqID, ltime, stime, seq, pkts, dur, and rate-related features (rate, proto_udp, proto_tcp), as well as a specific destination IP indicator (daddr_192.168.100.3). This aligns with the intuition that high-volume, high-rate traffic with particular timing patterns is characteristic of DDoS, but also confirms that the model exploits specific host information.

Overall, this implementation study shows that lightweight models such as Logistic Regression and Decision Trees can achieve perfect DDoS detection with zero false positives on BoT-IoT, supporting their suitability for deployment in resource-constrained IoT gateways—provided that the limitations of the dataset and the potential overfitting to the testbed environment are acknowledged. These points are explored further in the Critical Discussion section.

Critical Discussion & Best Practices

Comparison with existing literature

The models in this study achieve near-perfect performance on BoT-IoT: in the main experiment (Table 3.2) all three classifiers obtain 100% accuracy, precision, recall and F1 for DDoS, with FPR = 0 on normal traffic. Even in the larger, more imbalanced file used in preliminary experiments, Logistic Regression and Decision Tree still achieved perfect detection with zero false positives, and Random Forest produced only two false positives on ten normal test flows.

These results are broadly consistent with prior work on BoT-IoT. Several papers report accuracies above 99% using tree-based ensembles or deep learning, often after heavy feature selection and

resampling. However, those studies tend to present headline accuracy without giving much attention to false positives or to the complexity of the models in relation to IoT constraints. By contrast, this project explicitly reports FPR on normal traffic and compares a simple linear model (Logistic Regression) against more complex tree-based approaches using the *same* preprocessing pipeline and identical train–test splits. An interesting observation is that, unlike some studies where Random Forest significantly outperforms simpler models, here Logistic Regression and Decision Tree match or slightly outperform Random Forest. The confusion matrices in Figures 3.2–3.4, along with the identical metrics in Table 3.2, show that on this particular subset even a linear decision boundary is sufficient to perfectly separate DDoS from normal flows. This suggests that for BoT-IoT, the engineered flow features (packet/byte counts, rates, durations) and one-hot encoded metadata already create a representation where classes are almost linearly separable. Complex models therefore provide little additional benefit, which strengthens the case for lightweight methods in constrained environments.

Trade-offs for resource-constrained IoT networks

From an IoT deployment perspective, the main trade-offs are between detection quality, false positives, and resource usage. In terms of detection and FPR, all three models perform equally well on this dataset, so the differentiator becomes computational cost and implementation complexity.

Logistic Regression has the lowest training and inference cost: once the weights are learned, classifying a flow reduces to a dot product and a sigmoid. This is attractive for IoT gateways or edge routers with limited CPU and memory. Decision Trees are also relatively cheap at inference time, but can grow large and deep on more complex data. Random Forest, while still feasible at the edge in some scenarios, requires evaluating many trees per prediction and maintaining a larger model in memory. Given that no measurable accuracy or FPR advantage was observed for the more complex model, Logistic Regression (and to a lesser extent a shallow Decision Tree) appears to offer the best trade-off for this particular problem.

However, this conclusion is constrained by the characteristics of BoT-IoT. In more heterogeneous networks with noisier traffic, the decision boundary may be less linear, and ensembles or non-linear models might become necessary to maintain recall without increasing FPR. A practical best practice is therefore to start with the simplest model that meets detection/FPR requirements, and only escalate to more complex architectures if evaluation on diverse data shows clear benefits.

Emerging best practices for BoT-IoT-style DDoS detection

Several practical recommendations emerge from the implementation and results:

1. **Use flow-level features with minimal domain engineering.**
   The feature-importance analysis for Random Forest (Figure 3.8) shows that packet counts, byte rates, durations and protocol indicators dominate the decision process. This suggests that sophisticated handcrafted features are not strictly necessary for strong performance on BoT-IoT; standard flow statistics are sufficient.

2. **Avoid label leakage and keep the pipeline clean.**
   Removing attack, category and subcategory from the feature set was crucial to prevent the model from "cheating" by reading its own label. Clearly documenting this decision and the encoding of DDoS/Normal into numeric labels should be treated as a best practice for any IDS study on labelled datasets.

3. **Handle class imbalance deliberately and report FPR explicitly.**
   Both the original file (with 999,969 DDoS vs 31 Normal flows) and the smaller subset (1,044 vs 65) are imbalanced. Instead of relying on accuracy alone, this project focused on FPR for

normal traffic and visualised the confusion matrices. Future work on BoT-IoT should similarly report FPR and consider resampling (e.g. down sampling DDoS) or cost-sensitive learning to obtain more robust estimates.

4. **Prefer lightweight models when performance is comparable.**
   Given that Logistic Regression and Decision Tree matched Random Forest on all metrics, the extra computational cost of ensembles is hard to justify for this dataset. A reasonable best practice is to deploy a simple model (e.g. Logistic Regression) at the gateway and reserve heavier models (e.g. Random Forest or deep learning) for offline analysis or cloud-side correlation.

5. **Combine ML with basic network-level controls.**
   Even perfect detection in a testbed does not replace rate limiting, blacklisting and firewall rules. In practice, a good architecture would use the ML model to flag suspicious flows or hosts, then trigger network-level mitigation such as rate caps, connection throttling, or upstream filtering.

6. **Monitor and recalibrate over time.**
   Because IoT environments evolve, a one-off model trained on BoT-IoT is unlikely to remain optimal indefinitely. Periodic retraining on more recent traffic traces, and monitoring of FPR in production, are essential best practices to prevent model drift and alert fatigue.

Limitations and implications for generalisation

Despite the impressive metrics, there are important limitations that temper how far the findings can be generalised.

First, both experiments rely purely on BoT-IoT, a synthetic dataset with a limited set of IoT devices, fixed IP ranges and scripted attack patterns. The one-hot encoding of saddr and daddr allows models to exploit specific hosts (daddr_192.168.100.3 appears as a top feature), which is unlikely to be reliable in real deployments where IP addresses and device populations change. A more conservative design would aggregate at subnet level or avoid direct IP encoding altogether. Second, the normal traffic is extremely limited: only 31 normal flows in the large file and 65 in the smaller subset. As a result, the estimate of FPR is based on very few benign samples (10 in the first test split, 20 in the second). This means that FPR = 0 in Table 3.2 does not imply that the model would never generate false positives in practice; a single misclassified flow would significantly change the reported rate. A stronger evaluation would require more diverse and representative normal traffic, ideally from multiple IoT deployments. Third, only three classical supervised models were explored, all trained and tested on a static snapshot. No attempt was made to handle concept drift, adversarial adaptation, or online learning, which are critical in long-running IoT deployments. More advanced approaches—such as incremental learning, ensembles over time, or hybrid signature/ML systems—were beyond the scope of this portfolio but remain important directions for future work.

Overall, the critical takeaway is that BoT-IoT is a “friendly” dataset for supervised ML, and perfect scores on it should not be interpreted as proof that DDoS detection is “solved” in real IoT networks. Instead, the results show that, under controlled conditions and with appropriate preprocessing, simple models can achieve excellent detection and extremely low false positive rates, which is encouraging for resource-constrained environments. Turning this into a robust, deployable system will require validating the same models on additional datasets (e.g. other IoT or IIoT corpora), reducing reliance on host-specific features, and integrating the ML component into a broader network defence strategy.

Conclusion

This project ended up being as much about doing careful ML on security data as it was about "getting good results" At the start, my picture of IoT DDoS attacks was quite abstract, working with BoT-IoT made it concrete. Looking at flows rather than just packet captures showed how DDoS appears as extreme packet/byte rates, short durations and distinctive protocol/flag patterns. That helped me understand why flow-based IDS is popular in IoT a relatively small set of statistics can capture a lot of malicious behaviour. Most of the real work was in building a clean pipeline rather than tweaking algorithms. Early on, I hit issues such as accidentally treating attack/category as features, silent NaNs, and string columns ("e s" flags) breaking scaling. Fixing these forced me into better habits: explicitly separating labels from features, checking dtypes, confirming class mappings with crosstabs, and documenting every preprocessing step. I also saw first-hand how easy it is to produce "perfect" metrics for the wrong reasons if label leakage and imbalance are not handled properly. Class imbalance was especially eye-opening. The original BoT-IoT file (almost one million DDoS vs a few dozen Normal flows) showed how meaningless accuracy can be and how unstable FPR becomes when the benign class is tiny. Creating a smaller but more balanced subset improved the evaluation but also highlighted a fundamental limitation: BoT-IoT has very limited and homogeneous Normal traffic. That insight made me much more cautious in interpreting 100% scores and more willing to foreground limitations instead of treating them as an afterthought.

If I were to extend this work, I would validate the pipeline on another dataset (e.g. ToN-IoT or CICIoT2023), reduce reliance on one-hot encoded IP addresses by aggregating or removing them, and explore incremental learning to reflect how an IDS would need to adapt over time in a live IoT deployment. Against this background, the project answers the research question clearly. Using BoT-IoT flows labelled as DDoS and Normal, three supervised models—Logistic Regression, Decision Tree and Random Forest—were trained on a standardised feature set including packet/byte counts, rate and duration statistics, and one-hot encoded protocol, flag and address information. On the held-out test set (313 DDoS, 20 Normal), all three models achieved 100% accuracy, precision, recall and F1 for DDoS, with a false positive rate of 0 on Normal traffic. Preliminary experiments on a much larger, highly imbalanced file showed similarly strong performance, with only a couple of benign flows misclassified across hundreds of thousands of DDoS samples. These results suggest that, in the BoT-IoT setting, DDoS and Normal flows are highly separable and that lightweight models are sufficient: Logistic Regression and a simple Decision Tree matched Random Forest on every metric, despite being far cheaper to run. For resource-constrained IoT gateways this is encouraging, because it indicates that very strong detection with negligible false positives is achievable without resorting to heavy, complex models—at least when traffic resembles that seen in BoT-IoT.

However, the same results also underline important caveats. BoT-IoT is synthetic, its benign traffic is limited, and one-hot encoding of specific IP addresses encourages the models to memorise the testbed layout rather than learn generalisable patterns. The project therefore demonstrates what is possible under controlled conditions rather than claiming that ML-based DDoS detection in real IoT environments is solved.

In summary, the work shows that supervised machine learning can deliver excellent DDoS detection with extremely low false positives on BoT-IoT, and that simple models such as Logistic Regression are strong candidates for deployment in constrained IoT settings. At the same time, it highlights that sound preprocessing, honest handling of imbalance and explicit reporting of false positives are essential best practices if such systems are to be evaluated and deployed responsibly.

References

1. Sinha, S. (2024). *State of IoT 2024: Number of connected IoT devices growing 13% to 18.8 billion globally*. [online] IoT Analytics. Available at: https://iot-analytics.com/number-connected-iot-devices/.
2. Alosaimi, S. and Almutairi, S.M. (2023). An Intrusion Detection System Using BoT-IoT. *Applied Sciences*, [online] 13(9), p.5427. doi: https://doi.org/10.3390/app13095427
3. Ashraf, J., Raza, G.M., Kim, B.-S., Wahid, A. and Kim, H.-Y. (2025). Making a Real-Time IoT Network Intrusion-Detection System (INIDS) Using a Realistic BoT–IoT Dataset with Multiple Machine-Learning Classifiers. *Applied Sciences*, 15(4), p.2043. doi: https://doi.org/10.3390/app15042043
4. Pokhrel, S., Abbas, R. and Aryal, B. (2021). IoT Security: Botnet detection in IoT using Machine learning. *arXiv:2104.02231 [cs]*. [online] Available at: https://arxiv.org/abs/2104.02231
5. Kumari, P. and Jain, A.K. (2023). A Comprehensive Study of DDoS Attacks over IoT Network and Their Countermeasures. Computers & Security, 127, p.103096. doi: https://doi.org/10.1016/j.cose.2023.103096.