Evaluating the Impact of Feature Engineering Techniques on Supervised Machine Learning Algorithms for IoT Intrusion Detection Systems: A case Study on Mirai Botnet Attacks

MSc Academic Research Project

Cybersecurity

Dina Allam

Student ID: x23291583

School of Computing

National College of Ireland

Supervisor: Mustafa UI Raza

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	……Dina Allam……
Student ID:	……X23291583……
Programme: …… Cybersecurity	**Year:** …2025….
Module:	… Msc Research Project Configuration Manual
Lecturer:	……Mustafa Ul Raza ……
Submission Due Date:	……11 / 12/ 2025 ……
Project Title:	Evaluating the Impact of Feature Engineering Techniques on Supervised Machine Learning Algorithms for IoT Intrusion Detection Systems: A case Study on Mirai Botnet Attacks
Word Count:	……5382…… **Page Count:** ……23……

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary

Action.

Signature: ……Dina Allam……

Date: ……11/12/2025……..

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Evaluating the Impact of Feature Engineering Techniques on Supervised Machine Learning Algorithms for IoT Intrusion Detection Systems: A case Study on Mirai Botnet Attacks

Evaluating the Impact of Feature Engineering Techniques on Supervised Machine Learning Algorithms for IoT Intrusion Detection Systems: A case Study on Mirai Botnet Attacks

Abstract

In this new era, the integration of internet of things (IoT) devices in everyday environment became essential and demandable, this integration also broadens the global attack surface. This resulting with massive increase in IoT malware records between 2024 and 202. Long – standing attacks such as Mirai botnet continue to evolve, reinforcing the need of intrusion detection systems (IDS). This study evaluates the impact of different feature engineering techniques on the performance of supervised machine learning models in detecting Mirai botnet attacks, using recent Dataset. This study utilised most recent dataset named CIC IoT-DIAD 2024 that focusing on flow-based features of Mirai attacks. Three feature configurations were evaluated: original features, Autoencoder features and combined Autoencoder and PCA features on four supervised machine learning algorithms including Logistic regression, Random Forest, XGBoost and LightGBM. The results computed based on confusion matrix, showing that the best performance model were Random Forest and LightGBM using the original feature set with ~93% accuracy. These results highlighted the strength of ensemble machine learning models against linear models, it also showed that not always the feature engineering and reduction techniques enhance the performance. Future studies should explore more hand-crafted feature extraction techniques and deep learning models to enhance IDS detection rate.

Introduction

With the goal of connecting the unconnected, Internet of Things (IoT) devices are defined as physical objects with embedded hardware sensors, software cameras, or other technology that can transmit data (Hanes et al., 2017). These devices enable real-time monitoring and control while transmitting data across wired or wireless networks. These devices being integrated into wide range of smart. However, due to their limited processing power, storage and memory, IoT devices are particularly vulnerable to cyberattacks and often lack the capacity to implement strong security protocols and measures. Integrating different types of IoT devices into one system creates heterogeneous nodes, where different protocols, operating systems, level of security, and hardware significantly increase the challenge of securing the network and mitigating attacks (Altulaihan, Almaiah and Aljughaiman, 2024).

Integration of IoT devices into various systems has grown significantly by more than 15 % from 2020 and the end of 2023 (IoT Analytics, 2024). Recent statistics also indicate that the cyberattacks targeting IoT devices worldwide have increased by 32 million additional malware attacks between 2018 and 2022 (Statista, 2023). Although there are no publicly precise percentage breakdowns across IoT attack types, Zscaler (2023) reports that botnet attack types

like Mirai and GAFgyt-based contributing with almost 66% of the total malware payload amount. Mirai botnet is one of the most famous IoT botnets that specifically designed to compromise IoT devices by exploiting weak credentials resulting massive DDOS attacks controlled by Command-and-Control C2 server (Antonakakis *et al.*, 2017). Consequently, the need to develop reliable intrusion detection systems (IDS) for IoT becomes essential.

Generally, Intrusion Detection Systems (IDS) works on network layer where it analyses the packet to detect any deviation or up normality. Usually, it can operate at two levels: network and host level. The scope of IDS in host level is to detect any potential attack on single machine where at network level the scope expands to include all the moving traffic through the network (Mukhopadhyay et al., 2015). It employs different techniques to detect attacks, including signature-based, anomaly-based, hybrid-based approaches, and the integration of machine learning with IDS using heuristic-based methods (Mukhopadhyay et al., 2015). using heuristic-based methods enhances the detection of zero-day attacks, reduces false positives, enables adaptive learning and facilitates handling a large volume of data.

For the previously mentioned reasons, the number of research focused on developing IDS systems for IoT environments using machine learning has significantly increased.

Motivation and Gaps

The motivation behind developing IoT-related IDS is the rapid expand of IoT devices in modern networks, which reached approximately 15.9 billion devices by the end of 2023 and is forecasted to grow to 18 billion in 2024 (Statista, 2023). In addition to this general motivation, a more specific driver is the Mirai botnet and its historical and ongoing impact. Since October 2016, when a massive Layer-4 DDoS attack disrupted well-known platforms such as Twitter, Reddit, Amazon, and Netflix, Mirai and its variants have continued evolving (Antonakakis *et al.*, 2017). Recently, Cloudflare reported an attack in 2025 that reached volume of 5.6 terabits per second of traffic (Cloudflare, 2025).

A review of existing literature reveals several research gaps. First, the limited number of papers employing feature extraction methods, although some, such as Tyagi and Kumar (2021) reported high accuracy and low false positive rates despite relying on an outdated dataset. Second, many researchers relied on an outdated datasets such as IoT Network Intrusion Dataset, BoT-IoT dataset, IoTID20, and AWID.

These datasets do not adequately reflect the new attack trends and lack diversity in terms of IoT device representation. Finally, some researchers used not-specific IoT dataset such as NSL-KDD which is used in most studies conducted between 2015 and 2020 (Nugroho et al., 2020). Other commonly used datasets include KDD Cup 99, and CIC-IDS2017 which lacks IoT-specific traffic patterns and protocols such as MQTT, CoAP, or Zigbee, that are commonly observed in IoT networks. To address these gaps, this research study will focus on evaluating the performance of various machine learning algorithms using different feature selection and extraction methods on the most recent IoT dataset.

Research Question

How do various feature engineering methods impact the effectiveness of supervised machine learning-based intrusion detection systems, particularly in detecting Mirai botnet attack?

Objective

- Evaluate the impact of different feature engineering techniques including Autoencoders and PCA on detecting Merai botnet attacks using accuracy, precision, recall, F-score rate as evaluation matrix.
- Using updated dataset that includes latest attack trends such as different Merai botnet variants.

This document guides the reader through the literature review section, where related research is critically analysed, followed by the research methodology section, which presents the proposed method, the dataset used and its features, pre-processing phase, then feature engineering strategies and the machine learning models to be used for evaluation, after that the results where performance of the proposed model tested using confusion matrix mainly focusing on accuracy and F1-score, lastly the conclusions.

Literature review

While reviewing prior studies on machine learning for intrusion detection (IDS), it was observed that although many studies show promising results, relatively focused specifically on IoT environments. Furthermore, even fewer studies utilised updated IoT datasets or employed feature extraction techniques instead of feature selection. For instance, Verma and Ranga (2023) took an interesting approach, focusing on DoS attacks in IoT networks. The study employed binary classification, categorising network traffic as either normal or attack related. They assessed various ensemble-based classifiers such as Random Forest (RF), AdaBoost (AB), Gradient Boosted Machine (GBM), Extreme Gradient Boosting (XGB), Extremely Randomized Trees (ETC) and assessed simpler models like Classification and Regression Trees (CART), Multi-Layer Perceptron (MLP). Notably, it is the only reviewed study that evaluated classifier response time on a real IoT device Raspberry Pi. Unlike many studies that rely on a single dataset, the authors utilised three datasets CIDDS-001, UNSW-NB15, and NSL- KDD, to train the classifier and identify DOS attacks. While the inquiry acknowledged the potential negative impact of many features on the model and the importance of feature engineering, they did not detail their feature engineering process. The results highlighted RF for its lowest false positives, while CART for its fastest classification time, and XGB as most accurate. However, it lacks explicit feature engineering strategies. Moreover, it is worth mentioning that the NSL-KDD dataset is more general and does not reflect the IoT environment. it was primarily released as an update to the original KDD dataset from 1999, which had several issues affecting model performance. Consequently, the NSL-KDD dataset does not adequately present newer attack types found in modern systems (Tavallaee et al., 2009).

In contrast, a study by Liu et al. (2020) worked with a more recent dataset (IoT Network Intrusion Dataset) and investigated five classifiers, including Logistic Regression (LR), Support Vector Machine (SVM), K-Nearest Neighbours (KNN), Random Forest (RF), XGBoost .The dataset is specifically tailored for IoT environments and includes attacks such as the Mirai botnet. They selected seven features out of forty-two, focusing on attack–independent features, aiming to detect various attack types. Their study demonstrated that RF achieved the highest accuracy and overall performance, although it required significant computational resources. Conversely, KNN and XGBoost proved better for real-time detection. While their dataset includes IoT-specific attacks like Mirai botnets, it mainly covers basic IoT devices like cameras, smart home thermostats and assistants, routers and gateways (Kang et al., 2019). Given the rapid pace of technological advancement, this dataset is potentially missing newer threats and devices.

Another notable study by Alzahrani, Baabdullah and Rawat (2021) used a real-world dataset collected from a simple IoT setup using Arduino and NodeMCU-based ultrasonic sensors within a small IoT environment. While their RF and Gradient Boosting models achieved an impressive 99% accuracy through careful feature selection, these results may not be entirely reliable. However, the dataset adopted is limited in scope, lacks variety in attack scenarios. Consequently, while the dataset may be useful for testing IDS prototypes, it is insufficient as a benchmark for actual IoT security systems in production environments.

Alternatively, Tyagi and Kumar (2021) engaged feature extraction techniques while investigating various classifiers, including KNN, RF, Support Vector Machine (SVM), Logistic Regression (LR), Multilayer Perceptron (MLP), and Decision Tree (DT). The authors utilised the BoT-IoT dataset. Although this study highlighted feature extraction, it also used a limited scope scripted dataset that contains standard IP-based traffic, including limited attack types. The results highlighted RF with low false positive rates and high accuracy.

Moreover, a notable study by Gomez, Portela and Triana (2024) employed the most recent dataset released in 2024, CIC IOMT 2024, that includes a wide range of IoT devices focusing on the medical sector, various attack types and covers the recent attack vectors (Dadkhah et al., 2024). The authors investigated three models, including IForest, Local Outlier Factor (LOF) and Cluster-Based Local Outlier Factor (CBLOF). They also applied feature selection techniques, selecting seven out of forty-five features. Their results demonstrated that the LOF and CBLIOF achieved the highest accuracy, around 90%, but also the highest false positive rates, which negatively affect the effectiveness of the model.

On the other hand, there are some researchers that were interested in botnets. *Sharma and Babbar (2024)* proposed a model focusing on detecting Mirai botnet attacks, they utilised IoT-POT dataset, which was created 2015 as raw malware dataset, however it is continuously maintained and updated with malware binaries up to 2024 (YNU IoT Lab, 2025). Although the dataset looks promising and up to date, but it is unknown if the recent updates has included Mirai botnet or not. It is worth to mention that the performance of their model using Random Forest accompanied with selecting 12 features out of 22 that related to Mirai Attack resulting highest accuracy across other models such as Decision Tree, K-Nearest Neighbour, and

Logistic Regression. Although the accuracy of the model exceeds 99.5%, the Precision and Recall results are noticeably lower, which highlights the existing of false negative and false positive and giving probabilities of dataset imbalance.

Another approach worth to mention is to combine different machine learning algorithms in one ensemble voting classifier, this introduced by Alghamdi and Barsoum (2024). Their IDS proposed system focused on detecting Botnets using Gradient Boosting, Decision Tree, and Random Forest Classifier in one ensemble voting system after feature selection process and utilizing three datasets including NSL-KDD, CIC-IDS2018, and BoT-IoT. despite the results show promising detection performance when combine the three models comparing to using one model in detection, still there are risk when using the model for its recall scores with 88%. There are also concerns about how the trained model perform in more up to date attacks variants.

The following Table 1 summarise the reviewed studies, highlighting the utilised dataset, the feature engineering strategy and the attack category used.

Study name	Authors	Dataset used	Feature engineering	Study Focus
Machine Learning Based Intrusion Detection Systems for IoT Applications	Verma and Ranga	CIDDS-001, UNSW-NB15, and NSL-KDD	feature selection	DoS Attacks
Anomaly Detection on IoT Network Intrusion Using Machine Learning	Zhipeng Liu; Niraj Thapa; Addison Shaver	IoT Network Intrusion Dataset	feature selection	General attack types
Attacks and anomaly detection in IoT network using machine learning	Amani Alzahrani1,2,3(B), Tahani Baabdullah1,2,4(B), and Danda B. Rawat1	dataset collected from network logs using Ultrasonic Sensor with Arduino and NodeMCU	feature selection	General attack types
Attack and anomaly detection in IoT networks using supervised machine learning approaches	Tyagi and Kumar	BoT-IoT dataset	Feature Extraction	General attack types
intrusion Detection System for IoT using Anomaly Detection Techniques	Gomez, Portela and Triana	CIC IOMT 2024	feature selection	General attack types
IoT-POT: Machine Learning-based Detection of Mirai Botnet Attacks in IoT	Sharma and Babbar	IoT-POT Dataset	Manual Feature selection	Focusing on Mirai botnet
A Comprehensive IDs to Detect Botnet Attacks Using Machine Learning Techniques	Alghamdi and Barsoum	NSL-KDD, CIC-IDS2018, and BoT-IoT	Feature selection using Mutual Information (MI)	Focusing on Botnets

Table 1

Research Methodology

This study adopts an experimental research design and structured according to a systematic approach called Cross Industry Standard Process for Data Mining (CRISP-DM) (Hotz, 2018). to investigate the effectiveness of applying feature extraction methods with different Machine learning algorithms for detecting Mirai botnet attacks in IoT environments.

The CRISP-DM provides a series of well-defined phases. Each phase was carefully prepared and executed as follows and shown in Figure 1:

- The first phase, Business Understanding, was addressed in the literature review section involved evaluating current and past studies to identify the research gaps and proposing the research question.
- The second phase, Data Understanding, was carried out in two stages. The first stage involves evaluating the common datasets available in IoT-related intrusion detection and selecting the recent dataset. The second stage focused on understanding the dataset itself, its features, IoT device types included, and the overall volume and distribution of the data.
- The third phase, Data Preparation, involved handling the imbalanced data, cleaning the data by removing null values, and applying feature extraction techniques.
- The fourth phase, Modelling, involved the selection of machine learning classifiers.
- The fifth phase, Evaluation, focused on assessing the performance of the selected models to identify the most effective one based on the evaluation metrics selected.
- Finally, the Deployment Phase integrates all these stages into a complete framework to be prepared for execution and further experimentation.

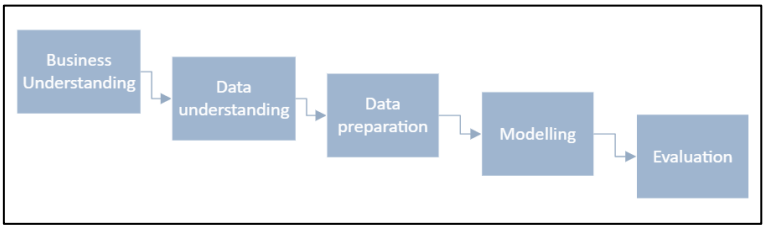


Figure 1- CRISP-DM phases

The research utilises the flow-based feature subset of the CIC IoT-DIAD 2024 dataset (AD_Flow-based-features) (Rabbani et al., 2024). This acts as the first layer of feature. Building on that, the study applies second layer of feature extraction using Autoencoders and Principal Component Analysis (PCA) to generate new high-level representation of the original features.

To evaluate the performance, different supervised machine learning algorithms including Logistic Regression, Random Forest, XGBoost and, LightGBM are applied to different feature sets:

- Original features
- Autoencoders bottleneck features
- Autoencoders bottleneck features + PCA dimensionally reduction features

The model architecture as shown in Figure 2.

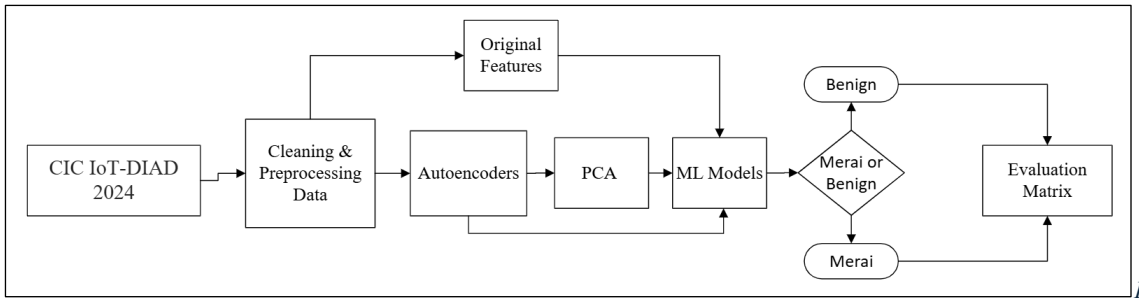


Figure 2- Methodology workflow

Dataset

The study utilises the CIC IoT-DIAD 2024 dataset, uses the flow-based features subset. This subset includes 85 network traffic features extracted from PCAP files using CICFlowMeter.

The records comprise bidirectional communication that are categorised according to the attack types, in this study the focus is limited to detecting Mirai botnet attack.

Design and Implementation Specifications

Experimental Environment

The experimental environment consisted of a machine equipped with an Intel® Core™ Ultra 9 processor. Configured with 32 GB of RAM and Intel® Arc™ integrated graphics unit featuring 128 graphics execution units (EUs). This configuration provided the sufficient computational resources for data preprocessing, feature extraction and training the machine learning models. The 64-bit operating system optimized the utilization of the processor and memory, enabling efficient model execution while reducing training time across all scenarios.

The Project was implemented in Python 3 using development environment (Google Colab). Several libraries were imported, including Numpy for handling mathematical calculations that were used in data preprocessing, Autoencoders and PCA. Pandas for handling tabular data (.csv) dataset files, and in preprocessing dataset. Seaborn and matplotlib, mainly used for its visualisation in confusion matrix and PCA variance graph.

Moreover, other libraries were imported to carry on the machine learning algorithm and specific preprocessing techniques such as class weighting, encoding labels and other techniques.

Data Preprocessing

The initial phase of the analysis involved comprehensive data preprocessing to ensure high-quality input for model training. The process began by loading the dataset and conducting exploratory analysis (EDA) to understand the data structure and identify potential issues.

Missing and infinite values were carefully handled to prevent computational errors during model training. These values replaced with median of each feature to ensure robustness against skewed data and don't influence by outliners while reducing the variance.

Non-numeric features that could not be meaningfully converted to numerical representations were removed from the feature set such as Flow ID, Src IP, Dst IP, Timestamp, as the chosen algorithms require numerical input.

After checking the class distribution of the dataset found that as shown in Figure 3 the Mirai samples is less than Benign samples.

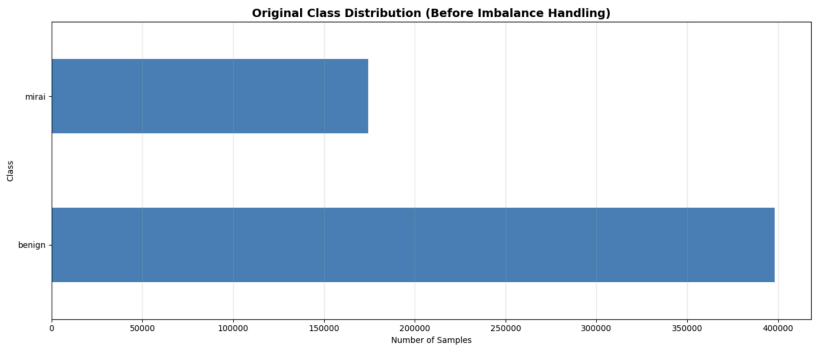


Figure 3

The imbalanced classes handled through class weights, on training set only. This technique penalizes misclassifications of the minority class more heavily by assigning higher weights to the minority class. The class weight was computed using the following Equation 1:

$$W_i = \frac{N}{k \times N_i}$$

Equation 1- Weights calculations

Where the total number of samples represented by N, the total number of classes represented by k and the number of samples in each class I represented by Ni. As result the computed weight for Benign class was 0.72, while the Merai class weighted higher for 1.64. this process will ensure the balance between classes without modifying the data distribution.

Feature Engineering Strategies

This study explored multiple feature engineering approaches to evaluate how different representations of the data influence model performance.

Original Features: The baseline configuration used the pre-processed original features without any form of dimensionality reduction or transformation. This approach preserves all available information but may include redundant or noisy features.

Autoencoder-Based Features: This technique is a type of artificial neural network designed to reconstruct the data, ensure noise reduction and extract features. While giving meaningful internal representation. With the architecture of 3 layers:

- The input layer (the Encoder), which compresses the input features,
- Hidden layer the bottleneck (latent space), which represents the informative compressed features, this layer will be set into 16,32, and 64 while examining the retained variance to check loss of relevant information.
- The output layer (the Decoder), which reconstruct the input data from the compressed representation.
- Number of layers for encoder and decoder: symmetric six layers, three for the encoder starting with 128 neurons, then 64 neuron and finally to the required dimension. Another three exact layers for the decoder as shown in Figure 4- Autoencoders used Architecture to perform gradual reduction without overfitting or underfitting problem.
- Batch size determines the sample size per training step, and it sets to 256 to mitigate noise and maintain acceptable training time.

Three separate Autoencoder neural networks were trained using varying bottleneck dimensions (16, 32, and 64). The compressed representations can capture non-linear relationships and potentially filter out noise while preserving the most important patterns in the data.

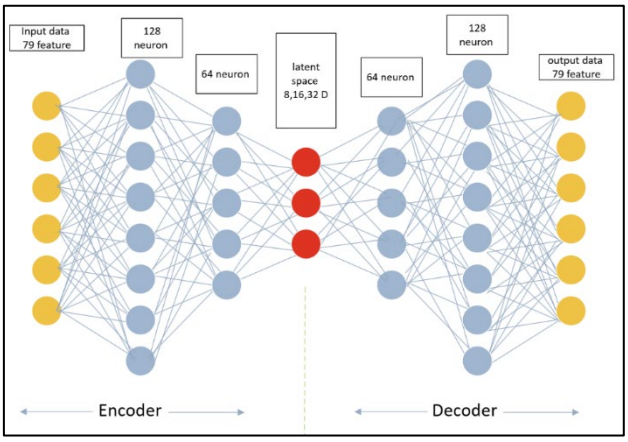


Figure 4- Autoencoders used Architecture

After completing the Autoencoders phase and checked its results that showed that the difference between the training loss and validation loss was relatively large. This is potentially could be due to overfitting. To address this problem, Autoencoders tuned by incorporating two techniques:

- dropout_rate=0.10: randomly dropping 10% of neurons during the training phase to prevent depending on specific neurons.
- l2_reg=1e-4: layer two regularization to penalise the large weights to keep the model smoother with generalised representation.

With these two features the gap between the validation loss and training loss is significantly reduced, along with improving in other attributes as shown in the following figure

PCA-Enhanced Autoencoder Features: principal component analysis (PCA) is used for linear dimensionally reduction for the dataset features while keeping the important information that is crucial in identifying the data points known as variance.

Building on the Autoencoder extracted features, Principal Component Analysis (PCA) was applied to further reduce dimensionality to 8, 16, and 32 dimensions as shown in Figure 6. This two-stage dimensionality reduction approach combines the non-linear feature learning capability of Autoencoders with the linear decorrelation properties of PCA, potentially yielding even more compact and informative feature representations.

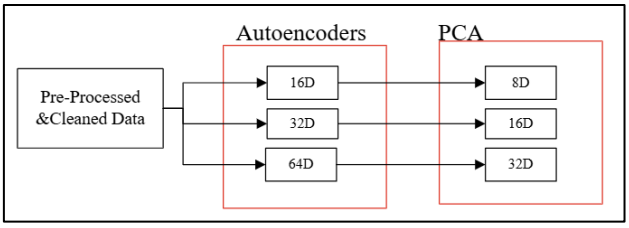


Figure 5- AE-PCA dimentionally reduction used techniques

The results obtained from PCA for dimensional reduction, shows that the variance retained better when reduction in more than 50% of the features as shown in **Error! Reference source not found.** and illustrated in the following Figure 7 where x-axis is the number to features after PCA and Y-axis is the percentage of retained variance.

Process	Features Before PCA	Features After PCA	Variance
AE16→ PCA8	16	8	91.65%
AE32→ PCA8	32	8	83.93%
AE32→ PCA16	32	16	94.90%
AE64 → PCA8	64	8	79.53%
AE64 → PCA16	64	16	91.28%
AE64 → PCA32	64	32	98.75%

Table 2

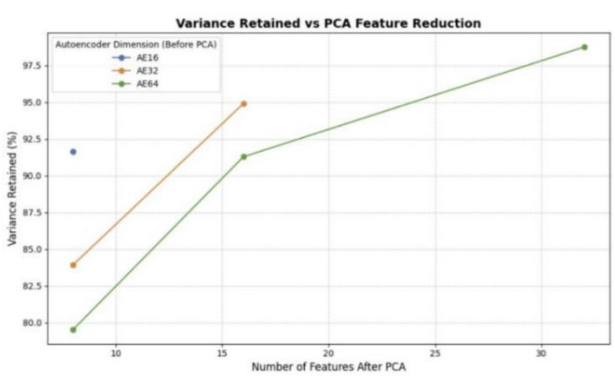


Figure 6

Machine learning Algorithms

Machine learning algorithms are generally classified into supervised, unsupervised and reinforcement learning categories. This study evaluated five classification algorithms representing different machine learning paradigms including Logistic Regression, Random Forest, XGBoost, and LightGBM.

The selection of these models was based on the differences between the concept of each model. Logistic regression was chosen to evaluate the performance of linear baseline model. However, Random Forest, XGBoost and LightGBM were chosen to evaluate the effectiveness of different ensemble learning algorithms, all of which are tree based but differ in their complexity, underlying mechanism and optimization strategies. This combination allows us to see how different feature sets perform against simple linear and complex non-linear classifier.

Logistic Regression

This model assumes linear relationship between the input features. Due to its characteristics, it acts as baseline to assess whether the problem requires complex non-linear modelling or can be solved with simpler approach. Depending on the logistic function as shown in Equation 2, the model predicts the probability of Mirai attack is a function of the set of given features Weather 0 or 1 as shown in the conceptual Figure 8.

$$P(Y=1) = \frac{1}{1 + e^{-(b_0 + b_1 X_{1i} + b_2 X_{2i} + \dots b_n X_{ni})}}$$

Equation 2- Logistic Function

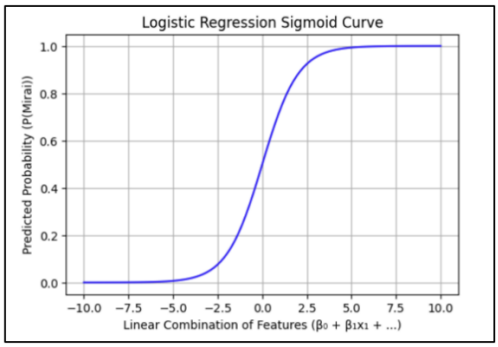


Figure 8

Random Forest (RF)

RF is an ensemble model based on the decision tree algorithm. It is a combination of multiple decision trees where each trained on randomly different subset of the training data and features. Each tree predicts its class by answering the main question on the root node, and from that node, multiple questions and conditions in branches until reaching a classification decision at the leaf node. RF uses bagging technique where it effectively reduces the Variance but without significantly increase the bias by calculate the votes to predict the output. Combining multiple decision trees improves accuracy. The n-estimator parameter is fine-tuned with 100 trees as shown in the following conceptual Figure 9.

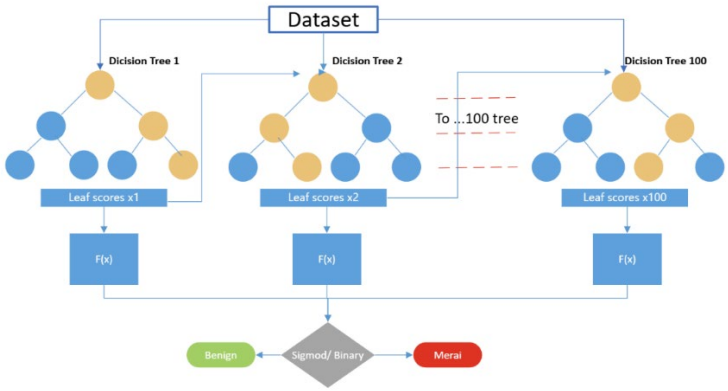


Figure 9 Random Forest conceptual diagram

XGBoost

Extreme Gradient Boosting is ensemble model gradient boosting algorithm that builds trees sequentially, with each tree attempting to correct errors made by previous trees. XGBoost is renowned for its performance in structured data problems. using Boosting technique where the new tree corrects its errors using the output from the last tree making this model more efficient as shown in the conceptual Figure 10.

XGBoost has multiple parameters that tuned to give better results as following:

- Learning-rate: set to '0.1' to avoid overfitting by controlling how much each tree contributes to the final prediction.
- Max-depth: set to '3' and refers to the maximum tree depth.
- N-estimators: number of trees set to '100' as RF algorithm.
- Eval-metric: set to 'logloss' as our current model dealing with binary classification Merai or benign.

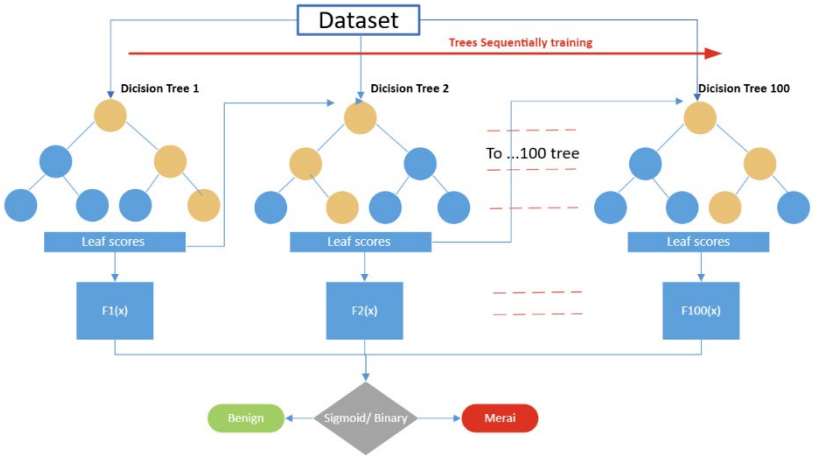


Figure 10-XGBoost Conceptual diagram

LightGBM

This is another boosting algorithm potentially known for its fast-training speed capability using histogram bins and exclusive feature bundling.

In contrast of the XGBoost and RF, LightGBM building the tree leaf wise not level wise allowing the model to reduce the loss and resulting better accuracy. Moreover, the max-depth of the tree set to 15 with 500 trees for deeper and more detailed relationships in the data.

Each model was trained on every feature set configuration, resulting in a comprehensive comparison matrix that reveals how different algorithms respond to various feature representations.

Evaluation Metrics

Model performance was assessed using two primary metrics:

- **Accuracy:** The proportion of correct predictions across all classes, providing an overall measure of model effectiveness.
- **F1 Score:** The harmonic mean of precision and recall, particularly valuable in imbalanced datasets as it balances the model's ability to correctly identify positive cases while avoiding false positives.

Additionally, confusion matrices were generated for each model-feature combination to provide detailed insight into classification patterns, including true positives, true negatives, false positives, and false negatives.

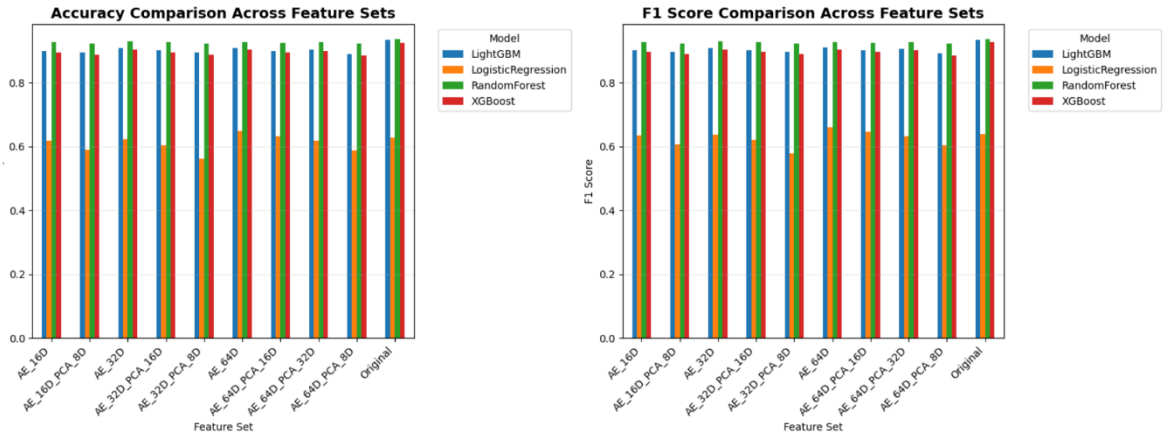
- **Precision:** is the percentage of exact positive predictions, either predicted as attack or normal, and calculated as below: true positive / (true positive + false positive).
- **False Positive:** is the percentage of classifying normal activity as an attack.
- **False Negative:** is the percentage of classifying up normal activity as normal (failing to identify an attack).
- **Recall:** it is the percentage of positive correctly predicted ones and calculated as true positive/ (false negative + true positive).

Results and Analysis

Overall performance comparison

After testing ten features sets across four models, the results revealed significant performance variations.

The accuracy and F1 scores comparison across all configurations are presented as shown in Figure 11.



Logistic Regression

Figure 11 revealed that Logistic Regression has the lowest overall performance across all the classifiers. This limitation could be due to its linear nature and its ability to handle complex relationships.

From the Table 2 below, the following observations can be made:

logistic regression		
feature set	Accuracy	F1
original	62.70%	63.94%
AE_64D	64.77%	66.04%
AE_32D_PCA_8D	56.30%	57.91%

Table 2

Using Autoencoders for feature extraction enhanced the model performance, especially with 64-dimentional.

Similarly, using the PCA reduction to 16 dimensions after Autoencoder stage with 64 dimensions achieved second highest score.

On the other hand, the model performance reduced when aggressive compression done- for instance, in case of compression from autoencoders (32D and 64D) to 8 dimensions using PCA.

This suggests that excessive dimensionally reduction have discarded discriminative features important in separating Benign from Merai samples.

Random Forest

Figure 11 revealed that Random Forest achieved the highest overall accuracy and F1- score across all used classifiers.

From the Table 3 below, the following observations can be made:

Random Forest

feature set	Accuracy	F1
original	93.53%	93.49%
Autoencoders	92.59% - 92.81%	92.54%- 92.76%
Autoencoders + PCA	92.17% - 92.6%	92.11%- 9.5%

Table 3

Random Forest achieved the highest overall accuracy and F1 score on the original features set. Using the Autoencoders or Autoencoders and PCA for feature extraction reduce the overall performance ~1%. The minimal variance across scores when feature extraction implemented showing the robustness the model.

XGBoost

Figure 11 revealed that XGBoost performed well using the original features and was better than Logistic Regression, but also the feature extraction techniques with different dimentions didn't enhance its performance as shown in Table 4

XGBoost		
feature set	Accuracy	F1
original	92.55%	92.61%
AE_64D	90.24%	90.34%
Autoencoders +PCA	89.50% -88.38%	89.62%-88.52%

Table 4

LightGBM

Figure 11 revealed that LightGBM scores are close to Random Forest scores when utilising the original features, while Autoencoders and PCA feature extraction methods reduced accuracy and F1-score ~2-3% as shown in Table 5 . This indicates that the classifier efficiently captured the nonlinear boundary between classes without additional feature extraction.

LightGBM		
feature set	Accuracy	F1
original	93.27%	93.31%
AE_64D	90.82%	90.92%
Autoencoders+ PCA	89.02%-90.32	89.15%-90.17%

Table 5

Best Performing Model by Feature Set

When examining the best model for each feature set independently, distinct patterns emerged, as shown in Table 6 and Figure 10.

Feature Set	*Model*	*Accuracy*	*F1 score*
AE-64D	Logistic regression	64.77%	66.04%
Original	Random Forest	93.53%	93.49%
Original	XGBoost	92.55%	92.61%
Original	LightGBM	93.27%	93.31%

Table 6

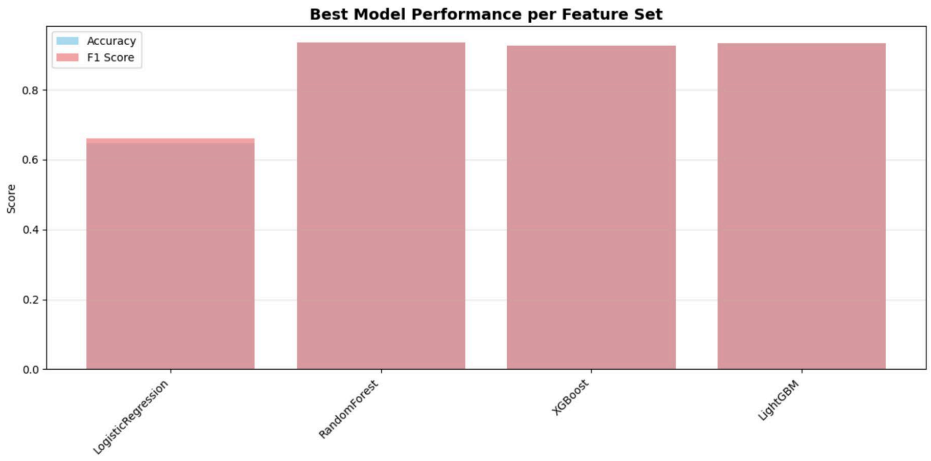


Figure 9

The original features consistently outperformed the engineered feature sets across most of the models, with three of the top four configurations using original feature set. This finding was somewhat surprising given the theoretical advantages of feature extraction and dimensionality reduction, but it suggests that the original features contained discriminative information that was lost during the compression process.

Confusion Matrix analysis

The following Figures 11-14 shows the confusion matrix for the best feature set in each classifier.

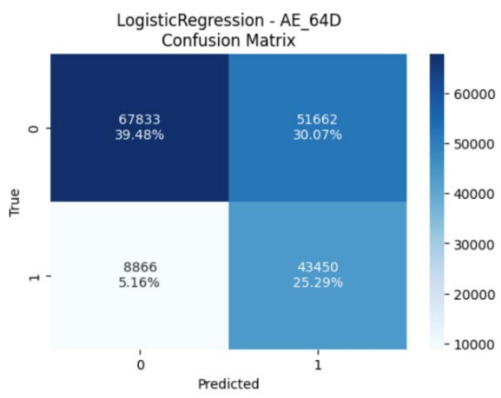


Figure 10- Logistic Regression

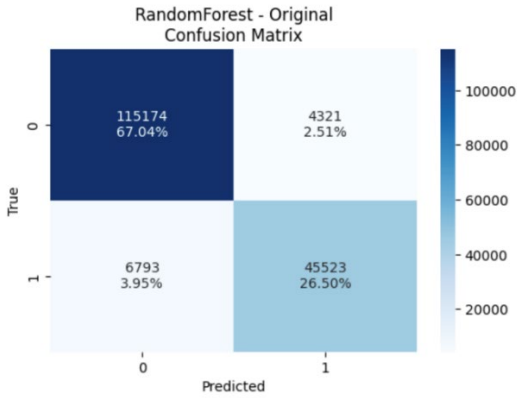


Figure 12- Random Forest

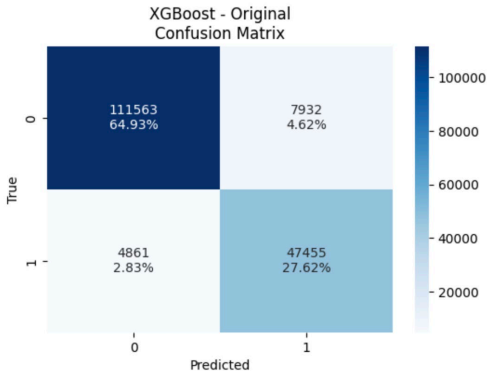


Figure 13- XGBoost

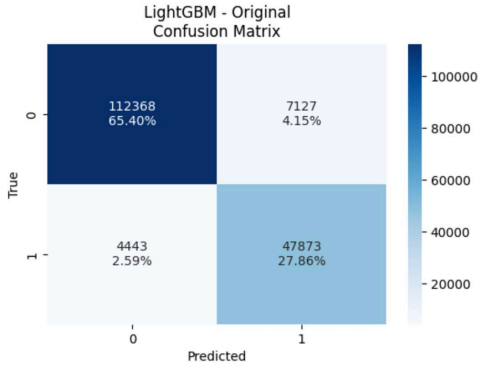


Figure 14- LightGBM

Key Findings and Insights

Several important patterns found during analysis:

Feature Representation Matters: Clear performance gap observed between original features and the compressed representations highlights that not all dimensionality reduction is beneficial. While Autoencoders can be powerful for certain tasks, they may struggle when discriminative information is distributed across many features in complex ways.

Ensemble Methods Excel: Random Forest, XGBoost, and LightGBM consistently outperformed linear models on the original features achieving accuracy levels around 92-93%. This suggests the decision boundaries in the anomaly detection problem are highly non-linear and benefit from the flexibility of tree-based ensembles.

Model-Feature Interactions: Interestingly, the performance ranking of models changed depending on the feature set. Logistic Regression, which performed modestly on original features (~62% accuracy), became the top performer on Autoencoder features. This suggests that the compressed representations may have linearized the feature space to some degree.

Minimal Performance Trade-offs: The best model (Random Forest on original features) achieved both the highest accuracy and nearly the highest F1 score (93.53% and 93.49% respectively), indicating balanced performance across both metrics without significant trade-offs between precision and recall.

Conclusions and future work

Based on the previous comprehensive evaluation, the recommended model is Random Forest classifier trained on original features for anomaly detection in production environments. This configuration achieved 93.53% accuracy and 93.49% F1 score, representing the best balance of performance and reliability among all tested approaches.

For organizations with computational constraints, LightGBM on original features offers a competitive alternative with slightly lower performance (93.27% accuracy) and it is viable option if training speed is a critical consideration.

The disappointing performance of Autoencoder-based features or the Autoencoders PCA based features suggests that future work should explore alternative feature engineering strategies, such as

hand-crafted domain-specific features, different neural network architectures, or hybrid approaches that combine original and learned features. Additionally, investigating why dimensionality reduction degraded performance could provide valuable insights for improving feature engineering methodologies.

In conclusion, this study demonstrates that careful model selection and feature engineering evaluation are essential for achieving optimal anomaly detection performance. While sophisticated feature learning techniques hold theoretical promise, empirical evaluation remains crucial for determining their practical effectiveness in specific problem domains. The Random Forest model's success on original features serves as a reminder that sometimes simpler approaches, when properly tuned and evaluated, outperform more complex alternatives.

References

Hanes, D., Salgueiro, G., Grossetete, P., Barton, R. & Henry, J. (2017) *IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things*. Indianapolis, IN: Cisco Press. Available at: https://learning.oreilly.com/library/view/iot-fundamentals-networking/9780134307091/ch01.html (Accessed: 24 May 2025).

Altūlaihan, E., Almaiah, M.A. & Aljughaiman, A. (2024) 'Anomaly detection IDS for detecting DoS attacks in IoT networks based on machine learning algorithms', *Sensors*, 24(2), p.713.

IoT Analytics (2024) 'State of IoT — Spring 2024: Number of connected IoT devices grows by 16% to 16.7 billion globally'. Available at: https://iot-analytics.com/number-connected-iot-devices/ (Accessed: 10 June 2025).

Statista (2023) 'Global annual number of IoT cyber attacks 2018–2022'. Available at: https://www.statista.com/statistics/1377569/worldwide-annual-internet-of-things-attacks/ (Accessed: 24 May 2025).

Zscaler (2023) 'Zscaler ThreatLabz finds a 400% increase in IoT and OT malware attacks year-over-year, underscoring need for better zero trust security to protect critical infrastructures', 24 October. Available at: https://www.zscaler.com/press/zscaler-threatlabz-finds-400-increase-iot-and-ot-malware-attacks-year-over-year-underscoring (Accessed: 16 August 2025).

Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., Durumeric, Z., Halderman, J.A., Invernizzi, L., Kallitsis, M. & Kumar, D. (2017) 'Understanding the Mirai botnet', in *26th USENIX Security Symposium (USENIX Security 17)*, pp. 1093–1110.

Mukhopadhyay, I., Gupta, K.S., Sen, D. & Gupta, P. (2015) 'Heuristic intrusion detection and prevention system', in *2015 International Conference and Workshop on Computing and Communication (IEMCON)*, October, pp. 1–7. IEEE.

Statista (2023) 'Number of Internet of Things (IoT) connected devices worldwide'. Available at: https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/ (Accessed: 24 August 2025).

Zscaler (2023) 'ThreatLabz 2023 Enterprise IoT and OT Threat Report'. *Zscaler*. Available at: https://info.zscaler.com/resources-industry-reports-threatlabz-2023-enterprise-iot-ot-threat-report (Accessed: 24 August 2025).

Cloudflare (2025) 'Record-breaking 5.6 Tbps DDoS attack and global DDoS trends for 2024 Q4', *Cloudflare Blog*, 21 January. Available at: https://blog.cloudflare.com/ddos-threat-report-for-2024-q4/ (Accessed: 24 August 2025).

Tyagi, H. & Kumar, R. (2021) 'Attack and anomaly detection in IoT networks using supervised machine learning approaches', *Revue d'Intelligence Artificielle*, 35(1).

Nugroho, E.P., Djatna, T., Sitanggang, I.S., Buono, A. & Hermadi, I. (2020) 'A review of intrusion detection system in IoT with machine learning approach: current and future research', in *2020 6th International Conference on Science in Information Technology (ICSITech)*, pp. 138–143. IEEE.

Podder, P., Mondal, M.R.H., Bharati, S. & Paul, P.K. (2020) 'Review on the security threats of Internet of Things', *International Journal of Computer Applications*, 176(41), pp. 37–45. doi:10.5120/ijca2020920548.

Verma, A. & Ranga, V. (2023) 'Machine learning-based intrusion detection systems for IoT applications', *arXiv preprint arXiv:2302.12452*. Available at: https://arxiv.org/abs/2302.12452 (Accessed: 30 May 2025).

Tavallaee, M., Bagheri, E., Lu, W. & Ghorbani, A.A. (2009) 'A detailed analysis of the KDD CUP 99 data set', in *Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA 2009)*, pp. 1–6.

Liu, Z., Thapa, N., Shaver, A., Roy, K., Yuan, X. & Khorsandroo, S. (2020) 'Anomaly detection on IoT network intrusion using machine learning', in *2020 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD)*, August, pp. 1–5. IEEE.

Kang, H., Ahn, D.H., Lee, G.M., Yoo, J.D., Park, K.H. & Kim, H.K. (2019) 'IoT network intrusion dataset'. Available at: https://dx.doi.org/10.21227/q70p-q449 (Accessed: 14 June 2025).

Alzahrani, A., Baabdullah, T. & Rawat, D.B. (2021) 'Attacks and anomaly detection in IoT network using machine learning', in *International Conference on Human–Computer Interaction*. Cham: Springer International Publishing, pp. 465–472.

Gomez, A.S., Portela, F.G. & Triana, O.A.D. (2024) 'Intrusion detection system for IoT using anomaly detection techniques', in *2024 IEEE VII Congreso Internacional en Inteligencia Ambiental, Ingeniería de Software y Salud Electrónica y Móvil (AmITIC)*, September, pp. 1–6. IEEE.

Sharma, A. & Babbar, H. (2024) 'IoT-POT: Machine learning-based detection of Mirai botnet attacks in IoT', in *2024 First International Conference on Innovations in Communications, Electrical and Computer Engineering (ICICEC)*, October, pp. 1–6. IEEE.

YNU IoT Lab (2025) 'Available dataset: IoTPOT and X-Pot malware binaries (Dataset-1)'. Yokohama National University. Available at: https://sec.ynu.codes/iot/available_datasets (Accessed: 6 September 2025).

Alghamdi, A. & Barsoum, A. (2024) 'A comprehensive IDS to detect botnet attacks using machine learning techniques', *Proceedings of the IEEE 3rd International Conference on Computing and Machine Intelligence (ICMI)*, April, pp. 1–6. IEEE.

Hotz, N. (2018) 'What is CRISP-DM?', *Data Science Process Alliance* [online]. Available at: https://www.datascience-pm.com/crisp-dm-2/ (Accessed: 5 October 2024).

Rabbani, M., Gui, J., Nejati, F., Zhou, Z., Kaniyamattam, A., Mirani, M., Piya, G., Opushnyev, I., Lu, R. & Ghorbani, A.A. (2024) 'Device identification and anomaly detection in IoT environments', *IEEE Internet of Things Journal*, December 2024.