National College of Ireland

Optimising Internet of Medical Things Intrusion Detection Performance using Machine Learning

MSc Research Project
MSc Cybersecurity

Charles Doherty
Student ID: x23287039

School of Computing
National College of Ireland

Supervisor: Ross Spelman

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Charles Doherty……
Student ID:	x23287039……
Programme:	MSc Cybersecurity…… **Year:** 2025……
Module:	Practicum Part 2 (Report)……
Lecturer:	Ross Spelman……
Submission Due Date:	15 September 2025……
Project Title:	Optimising Internet of Medical Things Intrusion Detection Performance using Machine Learning ……
Word Count:	7,716…… **Page Count:** 17 (excl. References, Appendix)

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: Charles Doherty……

Date: 25 August 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):	

Optimising Internet of Medical Things Intrusion Detection Performance using Machine Learning

Charles Doherty
Student ID: x23287039

Abstract

Internet of Medical Things (IoMT) networks are prime targets for cyber attacks due to their transmission and storage of sensitive health-related data. Performance advancement of lightweight machine learning (ML) algorithms is crucial to maximise the effectiveness of intrusion detection systems (IDS) in IoMT due to resource constraints. A review of related works noted an absence of detailed studies on how hyperparameter tuning efforts may improve the operation of an IDS specifically in an IoMT environment. Previous studies have been hindered by available datasets lacking diverse attack methods and devices specific to IoMT. This innovative research paired the assessment of multiple hyperparameter tuning methods with a recently published state-of-the-art dataset, the CICIoMT2024 dataset, to investigate potential performance enhancements. Inclusion of comprehensive details of input parameters provided for each tuning method is often omitted from research in the fields of hyperparameter tuning and intrusion detection. Publication of this information in this study enhances reproducibility of the work and benefits future studies. Selected models tested in this study were found to outperform those published by the authors of the CICIoMT2024 dataset for both 2-class (Binary: benign or malicious) and 6-class (Categorical: benign or category of attack) classification. Improvements in balancing performance across all categories was also obtained in results for Categorical classification in comparison to a recent publication. This work may be used as a foundation to improve generalisation of IoMT IDS performance by introducing additional datasets or expanding the number of categories used for multi-class classification efforts.

Keywords: Internet of Medical Things (IoMT), Intrusion Detection, Machine Learning (ML), Hyperparameter Tuning, Cybersecurity.

1 Introduction

The number of internet-connected devices in existence, collectively known as the Internet of Things (IoT), has steadily increased as advancements are made in technology. The use cases span multiple industries, with examples such as smart home devices, automated manufacturing technology and connected vehicles. These devices have also become widespread within the healthcare industry, where they are referred to as the Internet of Medical Things (IoMT). Devices are capable of carrying out functions including pulse and oxygen measurements, and network connectivity may be built into, for example, a blood pressure monitoring machine. Wearable technology containing remote sensors also play an important role in enhancing the

detail of patient records available to doctors when carrying out assessments, due to the readily available real-time metrics (Haleem et al., 2022). These devices may also send an alert if the condition of a patient deteriorates, and can assist with early detection of certain illnesses based on symptoms.

Due to the sensitivity of health-related data collected by IoMT devices, it is a frequent target for cyber attackers, motivated by the financial gains from extortion or sale of the data. An increase in the number of devices naturally leads to a larger attack surface for criminals to probe. Safeguarding patient data must be prioritised by health-related organisations. The research undertaken in this paper aims to optimise the attack detection capabilities of Machine Learning (ML) algorithms. A recently published IoMT dataset was specifically selected to enhance the defensive capabilities of healthcare providers, as the modern threat landscape is in a constant state of evolution due to changes in both device functionality and attack techniques. Incorporating varied attack types and state-of-the-art devices, this dataset provides a significant opportunity to train ML algorithms which can deliver real-world utility when deployed in an IoMT environment.

In addition to patient data being disclosed, attackers may also use IoMT device vulnerabilities as an entry point to systems where further collection of confidential intellectual property may occur (Javaid et al., 2023). As a result, successful breaches may result in an organisation facing significant financial and reputational damage. Reliance on legacy defences using perimeter-based methods, such as firewall rules to block external connections to an internal network, are increasingly less effective in the modern threat landscape. The non-traditional communication methods used by IoMT devices add further complexity when securing a network. As a result, a focus must be placed on improving the performance of intrusion detection systems (IDS). These systems continuously analyse traffic on the network and can populate dashboards with information or send alerts when potentially malicious traffic is discovered.

The following research question is proposed in order to address these issues faced by IoMT environments:

Can intrusion detection performance in an Internet of Medical Things (IoMT) environment be optimised through the use of multiple Machine Learning (ML) classification algorithms and hyperparameter tuning methods using a recent state-of-the-art IoMT dataset?

An analysis of the literature available strongly suggests a gap requiring further investigation of IoMT network data containing a broader range of device types and attack methods, enhancing ML model detection performance in the real world. This is covered in greater detail in Section 2. The CICIoMT2024 dataset (Dadkhah et al., 2024) presents an opportunity to address the shortcomings of several datasets used in relevant IoMT research up to now. For this reason, this dataset was selected for the research undertaken in this paper to improve ML attack detection capabilities. A distinct possibility noted during research was the prospect of

evaluating various ML algorithms using this chosen dataset with a specific focus on the effects of hyperparameter tuning methods. This selection was motivated by the absence of studies providing adequate detail on hyperparameter tuning in both the IoMT and the broader IoT intrusion detection body of research. It is suggested that with a thorough assessment of hyperparameter tuning methods the performance of algorithms may be measurably improved. Additionally, by documenting the hyperparameters space used and optimal parameters discovered during this research, it provides future researchers in this area with greater scope for reproducibility of results and may allow for time savings based on algorithm results using selected hyperparameters.

Section 2 covers related research in the field of IoMT intrusion detection, describing datasets available, common ML used and hyperparameter tuning within studies. The methodology followed when performing the research is outlined in Section 3. Both Design and Implementation of the research is described within Section 4. The Evaluation in Section 5 presents the notable results obtained, paired with discussion of the outcomes and relevant comparisons with alternate studies. Finally, the report concludes by outlining key findings and future possibilities for related work in Section 6.

2 Related Work

This section will present relevant research in related fields assessed prior to conducting the research for this project. The initial area of discussion gives an overview of the datasets previously made available for IoMT intrusion detection using ML, highlighting potential weaknesses and areas for improvement. Following this, the use of various ML algorithms was evaluated for suitability for the purpose of this research based on previous use. The emergence of papers being published that are using the CICIoMT2024 dataset is still in the early stages, but notable papers of significance are taken into account in relation to model selection. Key studies on hyperparameter tuning are covered for both IoMT and the broader field of IoT. A summary of findings concludes the review of the literature.

2.1 Datasets available for IoMT Intrusion Detection Research

IoMT is generally viewed as a subset of the broader field of IoT. There is no shortage in availability of IoT datasets, which often contain varied device and traffic types. De Keersmaeker et al. (2023) provide thorough documentation of publicly available IoT datasets in their 2023 survey paper. Many of these datasets documented are large in size, which assists real-world performance of intrusion detection in comparison to models which are trained on smaller datasets. Researchers in the area of IoMT have been seen to use general IoT datasets for research purposes. Gupta et al. (2020) used the UNSW-BOT-IoT and UNSW15 datasets, and even non-IoT network traffic datasets have been used, as seen with the use of the NSL-KDD dataset by Saheed and Arowolo (2021).

Despite use of these types of datasets for IoMT research, a contrasting viewpoint has been highlighted by many other researchers in this field. Across several papers in the current literature, a theme has emerged noting the importance of specialised IoMT datasets with

relevant devices being required to increase the effectiveness of IoMT intrusion detection. Firstly, many of the authors of the CICIoMT2024 dataset (Dadkhah et al., 2024) were involved in a review of IoT healthcare datasets (Neto et al., 2024), which identified the requirement for creation of a new IoMT dataset. This appeared to be the catalyst for the researchers to proceed with the creation of the CICIoMT2024 dataset, which will be discussed in greater detail later in this section. Another study by Hernandez-Jaimes et al. (2023) put forward the argument that detection performance is significantly impacted by unique characteristics and patterns within the training dataset during model creation. This may result in poor performance in a real IoMT environment for an IDS that was trained on IoT data not specifically related to healthcare devices. Their findings indicate that varied IoMT attacks and devices should be used to populate datasets in order to increase their utility.

An assessment was carried out into the IoMT dataset landscape in order to identify the most suitable dataset to choose for the proposed research. Current literature acknowledges recent advances in the quality and availability of IoMT datasets. Despite this, flaws still exist with several recently published datasets, with many failing to provide an adequate combination of size (number of instances) and required variety of both device traffic captured and attack methods in order to effectively train ML models for real-world applicability. Realistic testbed scenarios were used to capture the data for the BlueTack (Zubair et al., 2022) and WUSTL EHMS (Hady et al., 2020) datasets, but both are prohibitively small in size. WUSTL EHMS contains only 16,318 instances, and while BlueTack increases this number to 30,628, this is still considered very small in relation to training ML algorithms for intrusion detection capabilities, as they must be able to recognise a wide variety of malicious traffic. Ahmed et al. (2021) presented a larger dataset created with a complex IoMT testbed. Despite consisting of 111,207 rows, the ability to carry out effective ML model training using this dataset is severely limited due to it only containing nine features. The ICU dataset presented by Hussain et al. (2021) provides an improvement in size to 188,694 entries, with an increase in number of device types and types of attacks included. This is still not considered to be large in the context IoMT dataset requirements for ML, as the use of smaller datasets for training of models results in overfitting, reducing real-world performance due to training data lacking variety.

One of the most recent datasets published within this area has taken steps to address the issues detailed with previously discussed datasets. The data collected by Dadkhah et al. (2024) to populate their CICIoMT2024 dataset paves the way for new research opportunities in the field of IoMT intrusion detection. Improvements are clear in diversity of attack type and device content, which includes IoMT attacks that were not previously made available in other published datasets. The overall size of the dataset provides opportunities for adequate training of models, with the raw PCAP files containing network traffic captures consisting of over 8.7 million records. Accessibility of the data for research purposes is improved by the availability of CSV files created by the dataset authors, generated from processing the PCAP files. Classification results obtained by the dataset authors provide a baseline that can be used for comparison during research, and this will be further explored in Section 5.3 of this paper. This dataset provides a contemporary opportunity to optimise IoMT ML models to perform reliable intrusion detection.

2.2 Algorithm Selection

The task of intrusion detection in cybersecurity research is often explored using one or more types of Artificial Intelligence (AI), most often using Machine Learning (ML) or Deep Learning (DL). Both techniques use different mechanisms in their attempts to perform this task. ML learns from patterns found within training data fed into the model, while DL tries to mirror the mechanism of the human brain by processing and attempting to learn from the data presented to it (Hernandez-Jaimes et al., 2023). This study noted that DL algorithms require a much longer training time to perform classification tasks when compared with ML models.

Akar et al. (2025) used DL methods on the CICIoMT2024 dataset proposed for this research. A noted limitation of their work is the resource constraints faced within IoMT environments. In initial experiments carried out by the researchers, they found that ML models were effective in detecting intrusions, and that lightweight techniques should be further explored due to their increased efficiency and lower resource requirements. An additional recent study from Torre et al. (2025) also focused on DL methods, but discussed potential concerns over scalability and the increase in computational overhead required to implement the model in a real-world scenario. Further evidence on the likely model complexity issues accompanying the use of DL methods was found in the work carried out by Mezina et al. (2025), adding weight to the argument for favouring traditional ML models for a resource-friendly approach. As a result of these findings, paired with the domain knowledge that detecting intrusions in an IoMT environment is a time-sensitive exercise, this research focused heavily on the scope for improvement of ML models.

Efforts were made to identify if any ML models were shown to provide dependable high performance during classification tasks. Results from a survey paper on the use of AI in IoMT security applications (Hernandez-Jaimes et al., 2023) identified the Random Forest (RF) ML algorithm as an ideal candidate for use. This algorithm was the most consistent top performer across thirty-four papers, achieving the highest classification scores in nine of these papers. This provides evidence to back up the selection of the RF model for this research, especially when no other ML models provided standout performance across the body of research detailed in the survey. To further justify selection, RF also provided the best general performance in the research published by the authors of the CICIoMT2024 dataset (Dadkhah et al., 2024).

Due to the variety in results obtained using other ML models, a decision was made to base additional model selection on how often the models were utilised across existing research as opposed to the specific notable results obtained by the models. Ultimately, this resulted in the selection of three additional ML algorithms for use: Decision Tree (DT), K-Nearest Neighbours (KNN) and Support Vector Machine (SVM).

In an effort to provide an adequate contrast between the performance of ML methods with a DL model, the Multi-Layer Perceptron (MLP) algorithm was also selected to use during this research. This was done in order to assess if any potential performance gains over the ML models could be significant enough to overcome concerns around computational complexity

and resource requirements. The results of note obtained by MLP in intrusion detection research in a study within the IoT field (Li et al., 2024) provided reason for its selection.

Finally, it is important to note that the ability of each model selected to carry out both binary and multi-category classification was a factor in their selection, as the research necessitated testing of both scenarios. An analyst or automated system with the ability to recognise different threat types as part of an IDS may be able to take more relevant actions for a given situation, increasing the overall capability of the deployed solution.

2.3 Hyperparameter Tuning for IoT Intrusion Detection Models

While analysing research in IoMT intrusion detection to date, a notable gap was found in relation to hyperparameter tuning and the various methods that can be used to carry out this tuning. Dadkhah et al., authors of the CICIoMT2024 dataset (2024), specifically call out a potential for future work involving the refinement of hyperparameter settings to optimise the use of ML with their dataset. In order to explore how this may be achieved, we can assess studies that utilised hyperparameter tuning in IoMT and the broader field of IoT.

Abdelmoumin et al. (2022) performed research in relation to IoT intrusion detection, in which they discussed the possible use of three methods of hyperparameter tuning. These methods were grid search, random search, and Bayesian-based optimisation. However, only the random search method was implemented during their experiments, meaning that this work cannot be used to draw conclusions on the effectiveness of one method over another. Another study in this area was carried out by Li et al. (2024), with a focus on the comparison of feature engineering methods. While this study provides the settings for hyperparameters used for the models, it fails to elaborate on the reasons for selection of the hyperparameter values. Therefore, it is unclear whether testing or tuning was carried out in order to set these values, or merely the model defaults were used, leaving room for investigation into possible further optimisation of performance obtained. A grid search method was used along with the RF algorithm in work from Fernández Maimó et al. (2019). However, a very limited number of hyperparameters were chosen to be tuned. For example, only one RF hyperparameter was specified, namely the number of estimators, limiting the opportunity for a grid search to provide meaningful gains in classification metrics.

IoMT-specific studies related to ML hyperparameter tuning were also found when reviewing the research landscape. Binbusayyis et al. (2022) again perform grid search tuning, and provide both the input value range along with the best performing parameters discovered while tuning. However, a distinct gap is noted with the lack of comparison in classification results before and after using the tuned hyperparameters. As a result, it is not possible to clearly establish if the tuning process provided useful benefits to the algorithm performance. A random search hyperparameter tuning approach was found to be used in an emerging study which also selected the CICIoMT2024 dataset (Mezina et al., 2025). Unfortunately, a lack of input parameter bounds for the random search being specified in the paper affects the reproducibility of the work and how potential learnings could be applied to further studies. Specification of tuning inputs used for grid search was provided by Shaikh et al. (2025) in a recent paper, but

their work uses DL models as opposed to the lightweight ML algorithms chosen for this research, so no direct comparisons can be drawn.

2.4 Summary of Findings

From the findings within relevant studies outlined in this section, it is clear that a gap exists in the area of IoMT-specific studies for optimisation of intrusion detection using ML. This research specifically aims to add to the broader understanding of the potential positive performance effects of utilising hyperparameter tuning with ML, especially when paired with a state-of-the-art recently published IoMT dataset.

3 Overview of Research Methodology

The methodology followed during this research can be broken down into five main steps. An overview of the high-level tasks involved in each step is shown in Figure 1.

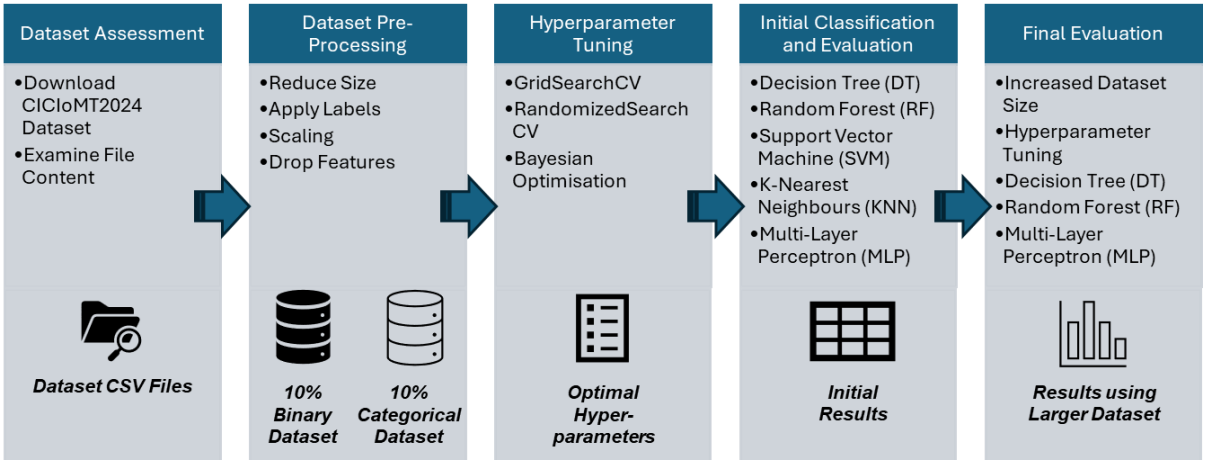

Figure 1: Research Methodology Overview

3.1 Dataset Assessment

The initial step involves obtaining and examining the source CICIoMT2024 dataset. The PCAP and CSV files were analysed, and a decision was made to use the CSV format for this research. Features provided in each CSV file were examined for suitability to perform ML classification tasks.

3.2 Dataset Pre-Processing

Initial tests were performed which determined that working with the full size of the source dataset was not feasible for the scope of this research project due to prohibitive algorithm processing times. As a result, the dataset was reduced in size by proportionally selecting data from the source CSV file set based on the class, and the size was further reduced to 10% of the previous reduction using stratified sampling. To ensure terminology when referring to original and processed datasets is consistent, a naming convention with details for each dataset being referred to is included in Table 1.

Table 1: Dataset Terminology and Descriptions

Terminology used in Report	Description	Method of Creation	Total Number of Instances	Malicious	Benign
CICIoMT2024 Dataset	Full CICIoMT2024 Dataset	Created by original dataset authors	8,775,013	8,544,674 (97.38%)	230,339 (2.62%)
Reduced Binary / Categorical Dataset	Subset of CICIoMT2024 Dataset	Proportional selection of data in Python based on category	1,057,414	827,075 (78.22%)	230,339 (21.78%)
10% Binary / Categorical Dataset	10% of previously Reduced Dataset	Stratified sampling of Reduced Dataset in Python code	105,742	82,708 (78.22%)	23,034 (21.78%)

Data was labelled as required, depending on the type of classification being performed. In the original research paper accompanying the dataset, three classification problems were considered: Binary (2-class: benign, malicious), Categorical (6-class: benign, spoofing, recon, MQTT, DoS, and DDoS), and Multi-Class (19-class). This research focuses on only Binary and Categorical classification, as hyperparameter tuning for Multi-Class was deemed to be too computationally complex to carry out within a realistic timeframe using available hardware. Additionally, in order to achieve adequate Multi-Class performance, a move towards DL methods would be needed, which provide greater performance but have limited real-world deployment possibilities due to IoMT computational resource constraints. Emerging DL results can be seen in work carried out by Akar et al. (2025). Scaling was performed to improve ML algorithm performance and unnecessary features were dropped.

3.3 Hyperparameter Tuning

Once data was imported from the CSV files and successfully pre-processed, hyperparameter tuning was carried out for both Binary and Categorical classification tasks using three different methods influenced by the literature overview presented in Section 2.3: grid search, random search, and Bayesian optimisation. All tuning input parameters along with each set of parameters discovered to be the best performing using each tuning method were recorded in the Appendix to this report, and were used in the following classification step of the process.

3.4 Classification and Evaluation

Five ML algorithms were chosen based on the rationale outlined in Section 2.2. For each of the five algorithms, and for both Binary and Categorical classification, a total of four runs were completed using different model hyperparameter settings. The parameters consisted of:

- Untuned (default) settings
- Optimal hyperparameter settings discovered using grid search
- Optimal hyperparameter settings discovered using random search
- Optimal hyperparameter settings discovered using Bayesian optimisation

All results were recorded using selected evaluation metrics outlined in Section 5. The choice of performance metrics was backed up by their widespread use in similar research (Dadkhah et al., 2024; Hernandez-Jaimes et al., 2023; Neto et al., 2023). The three algorithms demonstrating the highest results were chosen to use in the final evaluation.

3.5 Final Evaluation

Following the experiments carried out to find the three highest performing ML models using the 10% datasets, a final evaluation was conducted using the larger Reduced datasets to investigate if the optimal tuning methods found for the subset of the dataset may translate into an improvement in performance for the increased dataset size. The Decision Tree (DT), Random Forest (RF) and Multi-Layer Perceptron (MLP) algorithms were used for the final evaluation stage based on prior performance. Results were recorded and comparisons made to selected comparable research from the current literature.

4 Design and Implementation

4.1 Software Environment

All data processing and results gathering performed as part of this research was performed with Python code in Jupyter Notebook and using the Miniconda3 package manager. Key Libraries and packages used include NumPy and Pandas for working with the data, along with Sci-kit Learn for performing the majority of the ML algorithm implementation, with the exception of a standalone package used for Bayesian optimisation hyperparameter tuning.

4.2 Dataset Assessment and Pre-Processing

A visual representation of the dataset pre-processing stages can be seen in Figure 2. A large imbalance between the data classes was noted in relation to the source dataset. As per Table 1 in Section 3.2, 97.38% of the source dataset is malicious data, with the remainder benign in a Binary classification scenario. For Categorical data, the DDoS (~5.8m) and DoS (~2.2m) instances greatly outweighed the MQTT (~327k), Recon (~131k), Spoofing (~18k) and Benign (~230k) classes. To reduce this large imbalance, when creating the Reduced Binary / Categorical Dataset only 5% of DDoS, 10% of DoS, and 50% of MQTT instances were used in the data import, decreasing the malicious data representation to 78.22% for Binary classification. Categorical class-wise percentages for both the CICIoMT2024 dataset and the Reduced Dataset can be seen in Table 2, and the Reduced Dataset proportions were maintained for the 10% Dataset. This significant move towards a more balanced dataset may reduce the bias found in results from ML algorithms when training an IDS for IoMT intrusion detection (Hernandez-Jaimes et al., 2023).

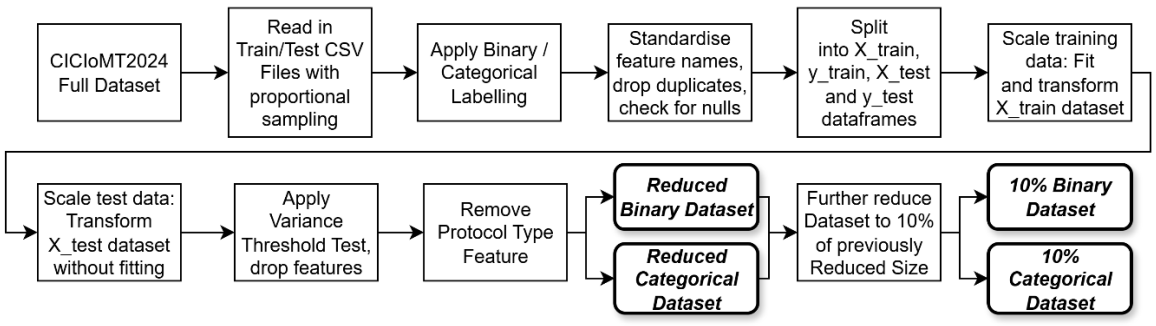

Figure 2: Dataset Pre-Processing to create Reduced and 10% Datasets

As the source CSV files do not contain labels for the instances, the process of labelling the instances is carried out during the import, as this is a requirement for supervised learning in ML (Akar et al., 2025). For Binary classification, a “malicious” feature is added, with a result of either False or True. For Categorical classification, a “category” feature is added, with values listed in Table 2. The category assignment is based on the names used in the original research paper (Dadkhah et al., 2024).

Table 2: Category Assignments and Category Representation within Datasets

Category Feature Value	Description	CICIoMT2024 Train Data Proportion	CICIoMT2024 Test Data Proportion	Reduced Dataset Train Data Proportion	Reduced Dataset Test Data Proportion
0	Benign	2.69%	2.33%	22.32%	19.40%
1	Spoofing	0.22%	0.11%	1.86%	0.90%
2	Recon	1.45%	1.71%	12.01%	14.27%
3	MQTT	3.67%	3.95%	15.22%	16.43%
4	TCP_IP-DoS	25.21%	25.81%	20.91%	21.49%
5	TCP_IP-DDoS	66.75%	66.09%	27.68%	27.51%

A full list of features within the CICIoMT2024 dataset can be found in the research paper accompanying the published dataset (Dadkhah et al., 2024). During the initial dataset examination, some anomalies were noted between the features contained within the CSV files and those listed in the paper. A total of seven features were found within the CSV dataset that were not listed as features in the paper: Srate, Drate, Duration, Magnitue (this is a typo of the word Magnitude, but listed here as per the CSV spelling), Radius, Covariance, and Weight. Additionally, the paper lists a “Time-To-Live” feature not found in the CSV files.

In the CICIoMT2024 paper, the authors refer to previous work on the CICIoT2023 Dataset (Neto et al., 2023) for their table of extracted features from the raw PCAP files. From this, we can obtain descriptions of the features which are present in the CSV files, but not mentioned directly within the paper, which are listed in Table 3. It can be inferred that the Duration feature in the CICIoMT2024 dataset is simply the "Time-to-Live" feature listed in their research paper, renamed to "Duration", based off the description provided for this feature.

Table 3: Features in CICIoMT2024 Dataset not detailed in associated research paper

Feature	Description
Srate	Rate of outbound packets transmission in a flow
Drate	Rate of inbound packets transmission in a flow
Magnitude	(Average of the lengths of incoming packets in the flow + average of the lengths of outgoing packets in the flow)^0.5
Radius	(Variance of the lengths of incoming packets in the flow + variance of the lengths of outgoing packets in the flow)^0.5
Covariance	Covariance of the lengths of incoming and outgoing packets
Weight	Number of incoming packets * Number of outgoing packets
Duration	Time-to-live

A notable issue was discovered with the “Protocol Type” feature in the CICIoMT2024 dataset. This is represented as a float value, described as "Mode of the protocols found in the window", with the window being a reference to the window-based sampling used to capture the data. In the CICIoT2023 dataset this value was an integer, which provided information on

the protocol type. For this research, a decision was made to drop this feature as the float values are merely a byproduct of various protocol types seen within the window, providing no meaningful data. The usage of protocol types are represented as separate features within the dataset: e.g. HTTP, HTTPS, Telnet, etc. A variance threshold test was also applied to the feature set, and this resulted in dropping the "Drate" feature as it had zero variance, therefore did not contribute any useful information in the dataset. This resulted in 43 features being used for model training and testing.

All feature names were standardised for consistency by converting all characters to lowercase, replaces spaces with underscores, and the "magnitue" typo was corrected to "magnitude". The training dataset was scaled to assist ML algorithm performance, with the same scaling then applied to the test dataset. Scaling was specifically noted as important to ensure all features could contribute to the ML model in research from Torre et al. (2025). Five scaling techniques were analysed by de Amorim et al. (2023) in relation to classification performance, and the Standard Scaler available in scikit-learn was subsequently chosen to apply scaling as it was seen to perform well within their study.

4.3 Hyperparameter Tuning Process

Hyperparameter tuning can help an ML model to increase accuracy, reduce overfitting or underfitting, increase the generalisation of a model and increase the efficiency of resource usage. The use of multiple methods to perform the tuning in this research is intended to maximise the opportunity to observe these refinements in model performance in comparison with models using the default parameters. The tuning strategies chosen use different methods to discover optimal parameters, with a brief outline given below:

- **Grid search (using GridSearchCV):** Checks every combination of all input parameters to discover the best performing set. *Drawback: Computationally expensive to run.*
- **Random search (using RandomizedSearchCV):** Performs a set number of tests using a random selection of parameters, where numeric inputs may be specified as a range instead of discrete values. *Drawback: May not randomly choose the optimal set of values.*
- **Bayesian optimisation (using bayesian-optimization):** Starts with random choices, but then uses a probabilistic model to work towards refining the output based on past results. *Drawback: Underlying probabilistic models are complex and may be time-consuming to execute.*

Each of the above methods requires a selection or range of hyperparameter values to be supplied as inputs. To aid reproducibility of this work, the full list of inputs for each tuning method, along with optimal hyperparameter settings found, is provided in the Appendix to this report. In a similar manner to research conducted by Binbusayyis et al. (2022), cross-validation was used to reduce overfitting of models against the training data provided, adapting the model better to real-world situations where it will process previously unseen data. While cross-validation helps to generalise the model, it does result in longer tuning times, so a balance must be struck. Tuning durations were recorded for comparative purposes, along with the optimal set of hyperparameters found using each of the three tuning processes. The tuning scoring metric was set to "f1" for Binary predictions, while "f1_macro" was used for categorical classifications as all classes are equally important to detect for this use case.

4.4 Classification Tasks

Experiments were carried out to capture the performance of the five chosen algorithms on both Binary and Categorical classification. To measure the effectiveness of tuning, results were obtained for the same algorithms and dataset for each classification type both with and without the hyperparameter tuning efforts in order to compare performance. An overview of the tuning and classification workflow is presented in Figure 3.

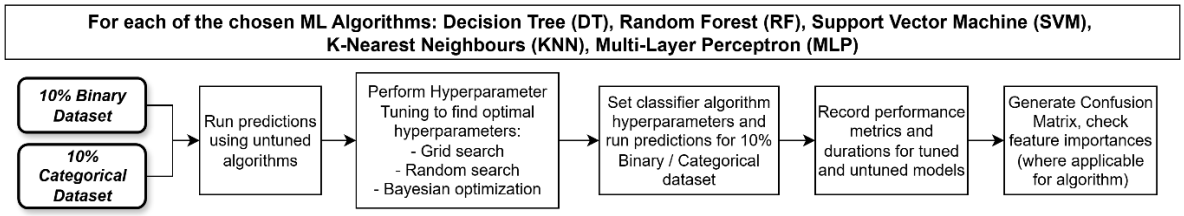

Figure 3: Tuning and Initial Classification Process

In each case, the algorithm was trained against the training dataset, then predictions were made on the target class of the instances within the test dataset. The source dataset was pre-divided by the creators into training and test data with an approximate 80% training and 20% test data split, a common proportion used in ML model assessment. This proportion was maintained when using both the 10% Datasets and the Reduced Datasets in this research. Results were recorded to collect information on the prediction performance of each algorithm for each set of hyperparameters provided, along with the durations for model predictions. A confusion matrix was generated following each classification to visualise the predictions, showing the numbers of true positives, true negatives, false positives and false negatives. This may provide additional insight on the specific misclassifications for a given model. Additionally, for the DT and RF algorithms, the feature importances were available to view as part of the built-in model implementation to show the most informative features used by the algorithm during its predictions. This can highlight if model predictions are too heavily reliant on one or several features, possibly a cause of overfitting a model on the training data, decreasing real-world performance capabilities.

For the final stage of the evaluation of the chosen algorithms, the Reduced Dataset was used in an attempt to assess tuned model performance against unseen data, as this dataset was significantly larger in size. Figure 4 details the process followed for this phase of the research. This process follows a similar path to the initial classification, but with fewer runs in the hyperparameter tuning stage and using only the DT, RF and MLP algorithms.

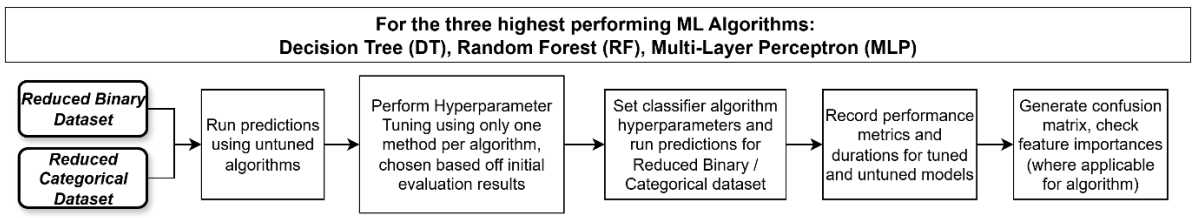

Figure 4: Final Evaluation Workflow

5 Evaluation

The performance metrics recorded for each set of predictions by a model are the Accuracy, Precision, Recall and F1-score. Accuracy should not be used as the primary measure of prediction performance for imbalanced datasets as this can provide misleading results by performing well on majority classes. Precision captures how often a positive prediction is made correctly, while recall tracks the proportion of the positive targets that were found. F1-score was selected as a primary metric as this takes both precision and recall scores into account. The weighted F1-score was chosen for overall Categorical scoring to reflect class imbalances. In this scenario, F1-score ensures a focus on detecting intrusions successfully for both Binary and Categorical classification tasks. Classifier Training Time and Inference Time are also captured.

5.1 Initial Classification and Evaluation

Binary classification using the 10% Binary dataset showed untuned versions of RF and KNN outperforming all tuned variations. Grid search resulted in the best performing DT model, Bayesian optimisation worked best for MLP, and random search optimal for SVM. For the 10% Categorical runs, SVM was only run in untuned mode and not tuned versions. This was done due to scope constraints, and SVM was dropped due to it performing worst in all Binary classification tasks and also in the untuned categorical run. Tuned algorithms provided superior categorical performance in comparison to untuned versions for all four remaining models: DT, RF, KNN and MLP. An overview of results obtained is presented in Table 4.

Table 4: Initial Classification Results using 10% Dataset

	Binary - 10% of Reduced Dataset: F1-score (%) for each Algorithm					**Categorical - 10% of Reduced Dataset: F1-score (%) for each Algorithm**				
Tuning Method	**DT**	**RF**	**KNN**	**MLP**	**SVM**	**DT**	**RF**	**KNN**	**MLP**	**SVM**
Untuned	98.25	**99.16**	96.99	97.39	96.84	98.11	98.93	88.04	77.86	75.25
Grid Search	98.39	98.58	96.89	97.04	96.45	98.52	98.94	88.19	79.34	N/A
Random Search	98.13	98.6	96.98	97.37	96.93	98.52	98.96	87.75	78.85	N/A
Bayesian Optimisation	91.3	98.65	96.49	97.57	89.54	98.45	**98.98**	88.31	79.75	N/A

Results Colour Key:

Optimal Task Performance

<0.25% from Optimal

>0.25% and <1% from Optimal

>1% from Optimal

5.2 Final Evaluation

Following SVM being dropped during the initial evaluation, KNN was also excluded from the final evaluation stage. Outside of SVM, it demonstrated the least capability in Binary classification, and the second from worst Categorical classification result using the 10% datasets. While Bayesian optimisation provided some good performance in the initial classification stage, it was only used for Binary tuning of MLP in the final evaluation, due to it providing the best MLP performance during 10% dataset testing. It was not used for other Binary models or any Categorical predictions due to its prohibitively slow runtime using the larger dataset, with Binary MLP tuning taking almost 24 hours to complete as documented in the Appendix. In these cases, either grid search or random search was used as a substitute tuning method, depending on performance during initial classification testing. Table 5 contains the results of classifications using the Reduced dataset.

The results obtained for Binary classification with the larger dataset do not follow those obtained from the 10% dataset. In a reverse of the 10% results, untuned DT and MLP outperformed the tuned versions. However, tuned RF provided the overall best binary performance for this test, highlighting the positive impact hyperparameter tuning can provide. However, significant model tuning times per the Appendix of this document must also be taken into account when comparing untuned and tuned performance differentials.

Weighted F1-score results for Categorical classification appear to show that untuned variants of DT and RF outperform tuned versions. However, class-wise results provide deeper insights into performance and may justify a preference for a tuned algorithm. This will be covered in detail in the discussion section to follow. Tuned MLP greatly outperforms the untuned version of the model, continuing to emphasise the benefits of tuning for categorical classification using MLP following the initial results of the 10% dataset testing.

Table 5: Final Classification Results using Reduced Dataset

	Binary - Reduced Dataset: F1-score for Algorithm			**Categorical - Reduced Dataset: F1-score for Algorithm**		
Tuning Method	**DT**	**RF**	**MLP**	**DT**	**RF**	**MLP**
Untuned	98.73	98.7	97.23	98.52	98.92	82.31
Grid Search	98.62	N/A	N/A	N/A	N/A	97.16
Random Search	N/A	98.88	N/A	98.5	98.78	N/A
Bayesian Optimisation	Not done - Time Constraints		97.09	Not done - Time Constraints		

Results Colour Key:
Optimal Task Performance
Not Performed

5.3 Discussion and Comparison

Firstly, it is important to note that findings did not remain consistent when variables such as the dataset size used and hyperparameter tuning type were modified during test runs. It can be inferred from these findings that one tuning method cannot be declared decisively superior to any other. An important learning is the importance of testing multiple methods that may provide an improvement in performance, and the most effective method cannot be reliably pre-determined. All three tuning methods each provided optimal hyperparameters for performance in different test cases. Also of note, untuned versions of algorithms at times outperformed tuned versions. This finding indicates that there must be an understanding of the tuning methods used, as applying them without rigorous testing may result in a degradation of model performance as opposed to boosting it.

As mentioned in section 5.2, untuned categorical classification results appeared to outperform tuned versions based off the weighted F1-scores. However, a look at the performance on each category of data (class-wise F1-score) elaborates on the surface level result, with results shown in Table 6 and Table 7. Firstly, for DT, we can see that the untuned model only outperforms the tuned DT for one class, which is Recon, showing that the tuned algorithm may provide more balanced performance in an IDS. Similarly for RF, we see that the tuned model performs better on all classes with the exception of Benign and Spoofing.

However, an increase in training and inference times accompany most tuned versions, with the exception of training time for the tuned DT being lower than untuned. When examining the confusion matrices in Figure 5 and Figure 5 for additional insight, it can be seen that the overall tuned result is impacted most significantly by misclassifications of Benign traffic as Spoofing (a false positive). The class imbalance in the dataset with a relatively small number of Spoofing samples may contribute to this poor performance (Mezina et al., 2025). Importantly, the untuned model classifies more malicious traffic as Benign (false negatives) when compared with the tuned model, a worst-case scenario for intrusion detection. With this knowledge, it may be reasonable to prioritise selection of the tuned version to maximise detection of attacks.

Table 6: Categorical Classification Results for Reduced Dataset (Part 1)

						Weighted Average			Benign		
Model	**Tuning**	**Tuning Time (s)**	**Training Time (s)**	**Inference Time (s)**	**Avg. Accuracy**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
DT	Untuned	N/A	28.603	0.101	98.39	98.73	98.39	98.52	97.95	95.12	96.51
DT	Random	364.7	19.297	0.171	98.49	98.52	98.49	98.50	97.70	95.43	96.55
RF	Untuned	N/A	355.304	2.717	98.9	98.95	98.90	98.92	97.39	97.65	97.52
RF	Random	28981.26	11208.3	15.272	98.69	98.92	98.69	98.78	97.56	96.44	97.00
MLP	Untuned	N/A	3296.96	0.232	82.09	82.7	82.09	82.31	94.73	93.18	93.95
MLP	Grid	40407.06	4197.56	1.464	96.79	97.65	96.79	97.16	94.14	91.99	93.05

Table 7: Categorical Classification Results for Reduced Dataset (Part 2)

		Spoofing			Recon			MQTT			TCP_IP-DoS			TCP_IP-DDoS		
Model	**Tuning**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
DT	Untuned	46.20	79.07	58.32	97.88	97.48	97.68	99.61	99.48	99.54	99.92	99.96	99.94	99.97	99.94	99.96
DT	Random	73.60	80.73	77.00	94.66	97.57	96.09	99.89	99.46	99.67	99.96	99.97	99.97	99.98	99.96	99.97
RF	Untuned	68.44	81.82	74.54	98.32	97.54	97.93	99.99	99.33	99.66	99.95	99.97	99.96	99.98	99.95	99.97
RF	Random	52.74	82.34	64.29	98.87	97.51	98.19	99.95	99.46	99.70	99.97	99.98	99.98	99.98	99.96	99.97
MLP	Untuned	28.61	60.84	38.91	98.20	95.05	96.60	99.78	98.47	99.12	65.55	61.07	63.23	71.15	74.89	72.98
MLP	Grid	23.78	54.87	33.18	98.14	94.97	96.53	99.80	98.18	98.98	99.79	99.12	99.46	99.34	99.84	99.59

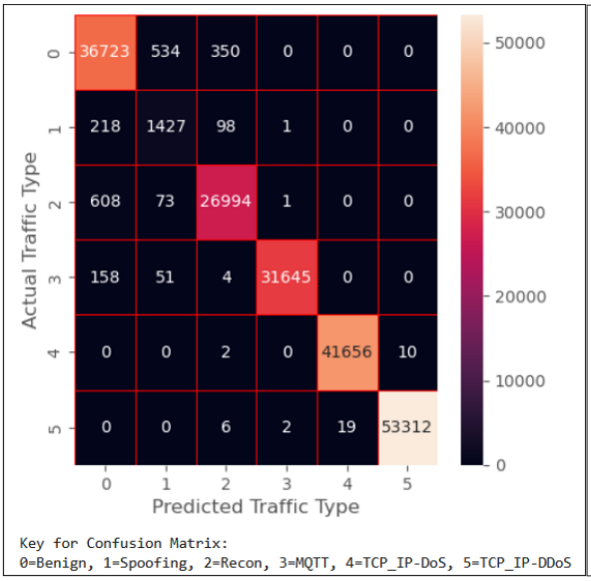

Figure 5: Untuned RF Confusion Matrix

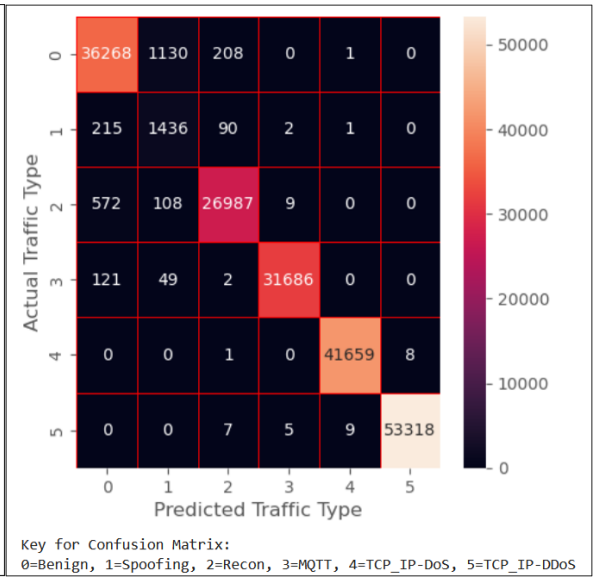

Figure 6: Random Search Tuned RF Confusion Matrix

Based on feature importances noted for the DT and RF models during final evaluation, in theory a number of features of lesser importance could be dropped from the feature set,

reducing computational complexity for predictions using the dataset. The maximum number of features with over 2% significance of any Binary DT or RF model was 15, with 18 the maximum number for Categorical testing. Feature reduction would allow the use of additional input features and more thorough cross validation for the tuning methods employed, potentially allowing for further performance improvements to be discovered via hyperparameter tuning.

In selecting works for comparison, the original paper which accompanied the dataset (Dadkhah et al., 2024) is the most obvious choice, as they carried out both 2-class and 6-class predictions. An improvement was obtained in maximum 2-class F1-score in this paper in comparison to Dadkhah et al., with 98.88% for the Reduced tuned dataset in contrast to their 96.1% achieved with RF. For 6-class (Categorical) classification, several of the models in Tables 6 & 7 can be seen to perform drastically better than the maximum 69.4% F1-score obtained by their Linear Regression model. The potential for this improvement using hyperparameter tuning was alluded to in the future work section of their study.

Rehman et al. (2025) also provide results using the same classes in their recent work. They report a 99.7% F1-score for Binary classification using the top 15 features with XGBoost, which outperforms the top result of 99.16% obtained in this research. However, their use of feature reduction means that results must be treated with caution, as over-simplification of the source data may result in improved model performance that may not directly translate to a real-world scenario. With the same features and model, they also report much higher class-wise Spoofing detection than any models tested in this work, with an F1-score of 94.2%. However, their Benign, Recon and MQTT scores fall short of those achieved by RF tuned using random search, as per Table 6 & 7, with DDoS and DoS classes being comparable in performance.

Finally, Mezina et al. (2025) discuss random search hyperparameter tuning while using four of the five algorithms used in this research, but only provide Binary and 19-class results, so no 6-class (Categorical) performance comparisons can be made. The number of different input parameters used for hyperparameter tuning was fewer in their study than in this work for the RF, DT, and MLP models, with the same number being used for KNN in both. For their Binary classification they balanced the classes between malicious and benign, which helps model performance, but by doing this it also resulted in a smaller dataset, with only 75,214 instances in the test set in comparison to 193,892 used in final evaluation during this research. They achieved an impressive F1-score of 99.86% with their DT model, but overfitting concerns exist due to the smaller training dataset of 192,732 instances, only 18.23% of the size of the dataset used in this study. Generalisation capabilities may be enhanced where models are trained on larger datasets.

6 Conclusion and Future Work

The stated goal of this research to optimise the performance of IoMT intrusion detection using various ML algorithms and hyperparameter tuning methods was partially achieved. Results obtained and compared with existing studies in Section 5 show clear improvements achieved in performance using hyperparameter tuning. However, improvements are not universal for all algorithms and tuning methods tested, and questions remain open regarding the balance of tuning efforts versus the achievable gains in model performance. The results obtained using the MLP DL algorithm do not appear to justify its selection ahead of traditional ML algorithms such as RF and DT, especially when considering the additional complexity of the DL model.

A significant contribution of this research to the greater body of work in this field is the thorough performance comparison of multiple hyperparameter tuning methods, paired with thorough documentation of both tuning input parameters and parameters found to provide optimal model performance. Positive results obtained have the potential to provide a pathway to improvement of future commercially deployed intrusion detection algorithms in an IoMT environment. Subsequent studies may expand on this through the use of the full CICIoMT2024 dataset and application of the methodology used in this paper to include 19-class classification tasks. However, this approach may require robust hardware resources to ensure feasibility, which may be at odds with the lightweight computational complexity requirements for deployment in an IoMT environment.

Feature reduction methods and the use of additional models could be used to expand the hyperparameter tuning tests performed here, with work from Rehman et al. (2025) supporting this theory. However, concern related to the use of fewer features resulting in overfitting of models to the training data must be addressed in this scenario. To combat this, multiple relevant datasets should be used in future efforts where possible, but difficulties related to their current availability in the field of IoMT have been discussed in Section 2.1 of this paper. Based on these findings, additional datasets may need to become available in order to further optimise models while adequately combatting overfitting to enhance real-world intrusion detection performance in the area of IoMT.

References

Abdelmoumin, G., Rawat, D.B. and Rahman, A. (2022) 'On the Performance of Machine Learning Models for Anomaly-Based Intelligent Intrusion Detection Systems for the Internet of Things', *IEEE Internet of Things Journal*, 9(6), pp. 4280–4290. https://doi.org/10.1109/JIOT.2021.3103829

Ahmed, M., Byreddy, S., Nutakki, A., Sikos, L.F. and Haskell-Dowland, P. (2021) 'ECU-IoHT: A dataset for analyzing cyberattacks in Internet of Health Things', *Ad Hoc Networks*, 122, pp. 102621. https://doi.org/10.1016/j.adhoc.2021.102621

Akar, G., Sahmoud, S., Onat, M., Cavusoglu, Ü. and Malondo, E. (2025) 'L2D2: A Novel LSTM Model for Multi-Class Intrusion Detection Systems in the Era of IoMT', *IEEE Access*, 13, pp. 7002–7013. https://doi.org/10.1109/ACCESS.2025.3526883

Binbusayyis, A., Alaskar, H., Vaiyapuri, T. and Dinesh, M. (2022) 'An investigation and comparison of machine learning approaches for intrusion detection in IoMT network', *J Supercomput*, 78(15), pp. 17403–17422. https://doi.org/10.1007/s11227-022-04568-3

Dadkhah, S., Neto, E.C.P., Ferreira, R., Molokwu, R.C., Sadeghi, S. and Ghorbani, A.A. (2024) 'CICIoMT2024: A benchmark dataset for multi-protocol security assessment in IoMT', *Internet of Things*, 28, pp. 101351. https://doi.org/10.1016/j.iot.2024.101351

de Amorim, L.B.V., Cavalcanti, G.D.C. and Cruz, R.M.O. (2023) 'The choice of scaling technique matters for classification performance', *Applied Soft Computing*, 133, pp. 109924. https://doi.org/10.1016/j.asoc.2022.109924

De Keersmaeker, F., Cao, Y., Ndonda, G.K. and Sadre, R. (2023) 'A Survey of Public IoT Datasets for Network Security Research', *IEEE Communications Surveys & Tutorials*, 25(3), pp. 1808–1840. https://doi.org/10.1109/COMST.2023.3288942

Fernández Maimó, L., Huertas Celdrán, A., Perales Gómez, Á.L., García Clemente, F.J., Weimer, J. and Lee, I. (2019) 'Intelligent and Dynamic Ransomware Spread Detection and Mitigation in Integrated Clinical Environments', *Sensors*, 19(5), pp. 1114. https://doi.org/10.3390/s19051114

Gupta, R., Tanwar, S., Tyagi, S. and Kumar, N. (2020) 'Machine Learning Models for Secure Data Analytics: A taxonomy and threat model', *Computer Communications*, 153, pp. 406–440. https://doi.org/10.1016/j.comcom.2020.02.008

Hady, A.A., Ghubaish, A., Salman, T., Unal, D. and Jain, R. (2020) 'Intrusion Detection System for Healthcare Systems Using Medical and Network Data: A Comparison Study', *IEEE Access*, 8, pp. 106576–106584. https://doi.org/10.1109/ACCESS.2020.3000421

Haleem, A., Javaid, M., Pratap Singh, R. and Suman, R. (2022) 'Medical 4.0 technologies for healthcare: Features, capabilities, and applications', *Internet of Things and Cyber-Physical Systems*, 2, pp. 12–30. https://doi.org/10.1016/j.iotcps.2022.04.001

Hernandez-Jaimes, M.L., Martinez-Cruz, A., Ramírez-Gutiérrez, K.A. and Feregrino-Uribe, C. (2023) 'Artificial intelligence for IoMT security: A review of intrusion detection systems, attacks, datasets and Cloud–Fog–Edge architectures', *Internet of Things*, 23, pp. 100887. https://doi.org/10.1016/j.iot.2023.100887

Hussain, F., Abbas, S.G., Shah, G.A., Pires, I.M., Fayyaz, U.U., Shahzad, F., Garcia, N.M. and Zdravevski, E. (2021) 'A Framework for Malicious Traffic Detection in IoT Healthcare Environment', *Sensors*, 21(9), pp. 3025. https://doi.org/10.3390/s21093025

Javaid, M., Haleem, A., Singh, R.P. and Suman, R. (2023) 'Towards insighting cybersecurity for healthcare domains: A comprehensive review of recent practices and trends', *Cyber Security and Applications*, 1, pp. 100016. https://doi.org/10.1016/j.csa.2023.100016

Li, J., Othman, M.S., Chen, H. and Yusuf, L.M. (2024) 'Optimizing IoT intrusion detection system: feature selection versus feature extraction in machine learning', *J Big Data*, 11(1), pp. 36. https://doi.org/10.1186/s40537-024-00892-y

Mezina, A., Nurmi, J. and Ometov, A. (2025) 'Novel Hybrid UNet++ and LSTM Model for Enhanced Attack Detection and Classification in IoMT Traffic', *IEEE Access*, 13, pp. 57589–57603. https://doi.org/10.1109/ACCESS.2025.3553966

Neto, E.C.P., Dadkhah, S., Ferreira, R., Zohourian, A., Lu, R. and Ghorbani, A.A. (2023) 'CICIoT2023: A Real-Time Dataset and Benchmark for Large-Scale Attacks in IoT Environment', *Sensors*, 23(13), pp. 5941. https://doi.org/10.3390/s23135941

Neto, E.C.P., Dadkhah, S., Sadeghi, S., Molyneaux, H. and Ghorbani, A.A. (2024) 'A review of Machine Learning (ML)-based IoT security in healthcare: A dataset perspective', *Computer Communications*, 213, pp. 61–77. https://doi.org/10.1016/j.comcom.2023.11.002

Rehman, M., Kalakoti, R. and Bahşi, H. (2025) 'Comprehensive Feature Selection for Machine Learning-Based Intrusion Detection in Healthcare IoMT Networks', in: *Proceedings of the 11th International Conference on Information Systems Security and Privacy*. Porto, Portugal, 20-22 February 2025, pp. 248–259. https://doi.org/10.5220/0013313600003899

Saheed, Y.K. and Arowolo, M.O. (2021) 'Efficient Cyber Attack Detection on the Internet of Medical Things-Smart Environment Based on Deep Recurrent Neural Network and Machine Learning Algorithms', *IEEE Access*, 9, pp. 161546–161554. https://doi.org/10.1109/ACCESS.2021.3128837

Shaikh, J.A., Wang, C., Sima, M.W.U., Arshad, M., Owais, M., Hassan, D.S.M., Alkanhel, R. and Muthanna, M.S.A. (2025) 'A deep Reinforcement learning-based robust Intrusion Detection System for securing IoMT Healthcare Networks', *Front. Med.*, 12, pp. 1524286. https://doi.org/10.3389/fmed.2025.1524286

Torre, D., Chennamaneni, A., Jo, J., Vyas, G. and Sabrsula, B. (2025) 'Toward Enhancing Privacy Preservation of a Federated Learning CNN Intrusion Detection System in IoT: Method and Empirical Study', *ACM Trans. Softw. Eng. Methodol.*, 34(2), pp. 53:1-53:48. https://doi.org/10.1145/3695998

Zubair, M., Ghubaish, A., Unal, D., Al-Ali, A., Reimann, T., Alinier, G., Hammoudeh, M. and Qadir, J. (2022) 'Secure Bluetooth Communication in Smart Healthcare Systems: A Novel Community Dataset and Intrusion Detection System', *Sensors*, 22(21), pp. 8280. https://doi.org/10.3390/s22218280

Appendix

Appendix Table 1 contains parameter settings for the hyperparameter tuning methods. The lists of input options and ranges for all hyperparameter tuning efforts performed are documented in Appendix Table 2. Finally, Appendix Table 3 consists of the optimal hyperparameter values found by performing each method for each algorithm listed, paired with the tuning time in each case.

Appendix Table 1: Hyperparameter tuning parameter settings

Dataset and Hyperparameter Tuning Method	Tuning Parameter Values - All Binary Algorithms	Tuning Parameter Values - All Categorical Algorithms
10% and Reduced Dataset Grid Search Parameters	scoring='f1',cv=4	scoring='f1_macro',cv=4
10% and Reduced Dataset Random Search Parameters	scoring='f1',cv=4[1],n_iter=50[2,3]	scoring='f1_macro',cv=4,n_iter=50[3]
10% and Reduced Dataset Bayesian Optimisation Parameters	scoring='f1',cv=3,init_points=5[4],n_iter=10[5]	scoring='f1_macro',cv=3,init_points=5[4],n_iter=10[5]
Exceptions to above settings: [1]3 for Reduced RF. [2]5 for 10% SVM (Binary only). [3]10 for Reduced RF. [4]10 for 10% DT. [5]30 for 10% DT.		

Appendix Table 2: Input settings for Hyperparameter Tuning

Hyperparameter Name	Grid Search Hyperparameters	Random Search Hyperparameters	Bayesian Optimisation Hyperparameter Bounds
Decision Tree (DT) Hyperparameters (Binary / Categorical)			
criterion	'gini', 'entropy'	'gini', 'entropy'	N/A
max_depth	10,20	np.linspace(10,100,dtype=int)	(1, 100)
min_samples_split	2,5,15	np.linspace(2,15,dtype=int)	(2, 25)
min_samples_leaf	1,4	np.linspace(1,6,dtype=int)	(0.1, 0.999)
max_features	'sqrt',None	np.linspace(0.1,0.999)	(1,4)
Random Forest (RF) Hyperparameters (Binary / Categorical)			
n_estimators	100,500	np.linspace(100,1000,dtype=int)	(10,1000)
max_depth	10,20	np.linspace(10,100,dtype=int)	(1, 100)
min_samples_split	2,6	np.linspace(2,6,dtype=int)	(2, 25)
min_samples_leaf	1,4	np.linspace(1,4,dtype=int)	(0.1, 0.999)
max_features	'sqrt',None	np.linspace(0.1,0.999)	(1,4)
Multi-Layer Perceptron (MLP) Hyperparameters (Binary / Categorical)			
max_iter	50,200	np.linspace(50,200,dtype=int)	(50,200)
learning_rate_init	0.001,0.005	np.linspace(0.001,0.01)	(0.001,0.01)

Hyperparameter Name	Grid Search Hyperparameters	Random Search Hyperparameters	Bayesian Optimisation Hyperparameter Bounds
learning_rate	'constant','adaptive'	'constant','adaptive'	N/A
activation	'tanh', 'relu'	'tanh', 'relu'	N/A
K-Nearest Neighbours (KNN) Hyperparameters (10% Dataset only, Binary / Categorical)			
n_neighbors	93,193,293,393	np.linspace(93,493,dtype=int)	(10,500)
weights	'uniform','distance'	'uniform','distance'	N/A
metric	'minkowski','euclidian', 'manhattan'	'minkowski','euclidian', 'manhattan'	N/A
Support Vector Machine (SVM) Hyperparameters (10% Dataset only, Binary only)			
kernel	'rbf','linear'	'rbf','linear'	N/A
C	0.0001,1,10	np.linspace(0.0001,10)	(0.0001,10)
gamma	1,10,100	np.linspace(1,100)	(1,100)

Appendix Table 3: Optimal Hyperparameters discovered during tuning process

Initial Evaluation - 10% Dataset			
Algorithm	**Dataset and Tuning Type**	**Tuning time (s)**	**Optimal Parameters Found**
DT	Binary Grid Search	29.11	{'criterion': 'gini', 'max_depth': 10, 'max_features': None, 'min_samples_leaf': 1, 'min_samples_split': 2}
DT	Binary Random Search	32.79	{'min_samples_split': np.int64(11), 'min_samples_leaf': np.int64(5), 'max_features': np.float64(0.889), 'max_depth': np.int64(74), 'criterion': 'gini'}
DT	Binary Bayesian Optimisation	67.56	{'max_depth': np.float64(1.0), 'max_features': np.float64(0.1), 'min_samples_leaf': np.float64(2.958), 'min_samples_split': np.float64(15.923)}
RF	Binary Grid Search	2685.47	{'max_depth': 20, 'max_features': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 100}
RF	Binary Random Search	10860.62	{'n_estimators': np.int64(871), 'min_samples_split': np.int64(4), 'min_samples_leaf': np.int64(2), 'max_features': np.float64(0.522), 'max_depth': np.int64(77)}
RF	Binary Bayesian Optimisation	9890.75	{'max_depth': np.float64(54.311), 'max_features': np.float64(0.551), 'min_samples_leaf': np.float64(1.216), 'min_samples_split': np.float64(8.174), 'n_estimators': np.float64(504.884)}
KNN	Binary Grid Search	687.67	{'metric': 'minkowski', 'n_neighbors': 93, 'weights': 'distance'}
KNN	Binary Random Search	1355.81	{'weights': 'distance', 'n_neighbors': np.int64(117), 'metric': 'manhattan'}
KNN	Binary Bayesian Optimisation	331.1	{'n_neighbors': np.float64(499.920)}
MLP	Binary Grid Search	1284.78	{'activation': 'tanh', 'learning_rate': 'constant', 'learning_rate_init': 0.001, 'max_iter': 200}
MLP	Binary Random Search	2753.53	{'max_iter': np.int64(98), 'learning_rate_init': np.float64(0.00118), 'learning_rate': 'adaptive', 'activation': 'relu'}
MLP	Binary Bayesian Optimisation	3942.07	{'learning_rate_init': np.float64(0.00183), 'max_iter': np.float64(158.518)}
SVM	Binary Grid Search	13394.44	{'C': 10, 'gamma': 1, 'kernel': 'rbf'}
SVM	Binary Random Search	8275.61	{'kernel': 'linear', 'gamma': np.float64(81.816), 'C': np.float64(5.306)}
SVM	Binary Bayesian Optimisation	35780.13	{'C': np.float64(2.730), 'gamma': np.float64(63.161)}

Initial Evaluation - 10% Dataset			
Algorithm	**Dataset and Tuning Type**	**Tuning time (s)**	**Optimal Parameters Found**
DT	Categorical Grid Search	35.67	{'criterion': 'entropy', 'max_depth': 20, 'max_features': None, 'min_samples_leaf': 4, 'min_samples_split': 15}
DT	Categorical Random Search	37.06	{'min_samples_split': np.int64(11), 'min_samples_leaf': np.int64(4), 'max_features': np.float64(0.944), 'max_depth': np.int64(15), 'criterion': 'entropy'}
DT	Categorical Bayesian Optimisation	102.71	{'max_depth': np.float64(100.0), 'max_features': np.float64(0.999), 'min_samples_leaf': np.float64(4.0), 'min_samples_split': np.float64(14.163)}
RF	Categorical Grid Search	4166.31	{'max_depth': 20, 'max_features': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 500}
RF	Categorical Random Search	10860.62	{'n_estimators': np.int64(871), 'min_samples_split': np.int64(4), 'min_samples_leaf': np.int64(2), 'max_features': np.float64(0.522), 'max_depth': np.int64(77)}
RF	Categorical Bayesian Optimisation	10354.46	{'max_depth': np.float64(54.311), 'max_features': np.float64(0.551), 'min_samples_leaf': np.float64(1.216), 'min_samples_split': np.float64(8.174), 'n_estimators': np.float64(504.884)}
KNN	Categorical Grid Search	680.61	{'metric': 'manhattan', 'n_neighbors': 93, 'weights': 'distance'}
KNN	Categorical Random Search	1334.96	{'weights': 'distance', 'n_neighbors': np.int64(117), 'metric': 'manhattan'}
KNN	Categorical Bayesian Optimisation	140.55	{'n_neighbors': np.float64(10.0)}
MLP	Categorical Grid Search	1562.2	{'activation': 'tanh', 'learning_rate': 'constant', 'learning_rate_init': 0.001, 'max_iter': 200}
MLP	Categorical Random Search	3306.99	{'max_iter': np.int64(108), 'learning_rate_init': np.float64(0.00431), 'learning_rate': 'constant', 'activation': 'tanh'}
MLP	Categorical Bayesian Optimisation	4029.7	{'learning_rate_init': np.float64(0.00396), 'max_iter': np.float64(67.296)}
Final Evaluation - Reduced Dataset			
Algorithm	**Dataset and Tuning Type**	**Tuning time (s)**	**Optimal Parameters Found**
DT	Binary Grid Search	256.39	{'criterion': 'entropy', 'max_depth': 20, 'max_features': None, 'min_samples_leaf': 4, 'min_samples_split': 15}
RF	Binary Random Search	31136.7	{'n_estimators': np.int64(871), 'min_samples_split': np.int64(5), 'min_samples_leaf': np.int64(2), 'max_features': np.float64(0.705), 'max_depth': np.int64(32)}
MLP	Binary Bayesian Optimisation	86041.94	{'learning_rate_init': np.float64(0.00169), 'max_iter': np.float64(166.988)}
DT	Categorical Random Search	364.7	{'min_samples_split': np.int64(10), 'min_samples_leaf': np.int64(4), 'max_features': np.float64(0.834), 'max_depth': np.int64(57), 'criterion': 'entropy'}
RF	Categorical Random Search	28981.26	{'n_estimators': np.int64(779), 'min_samples_split': np.int64(2), 'min_samples_leaf': np.int64(1), 'max_features': np.float64(0.889), 'max_depth': np.int64(81)}
MLP	Categorical Grid Search	40407.06	{'activation': 'tanh', 'learning_rate': 'constant', 'learning_rate_init': 0.005, 'max_iter': 200}