Received 2 February 2024, accepted 10 February 2024, date of publication 19 February 2024, date of current version 23 February 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3367113

RESEARCH ARTICLE

Protecting Smart-Home IoT Devices From MQTT Attacks: An Empirical Study of ML-Based IDS

RANA ALASMARI[1] **AND AREEJ ALHOGAIL**[2]
[1]College of Computer and Information Sciences, King Saud University, Riyadh 11362, Saudi Arabia
[2]STCs Artificial Intelligence Chair, Department of Information Systems, College of Computer and Information Sciences, King Saud University, Riyadh 11362, Saudi Arabia

Corresponding author: Areej Alhogail (aalhogail@ksu.edu.sa)

This work was supported by the Deanship of Scientific Research, King Saud University for funding through the Vice Deanship of Scientific Research Chair.

ABSTRACT Smart homes are becoming increasingly popular worldwide, and they are mainly based on Internet of Things (IoT) technologies to enable their functionality. However, because IoT devices have limited computing power and resources, implementing strong security measures is difficult, making the use of intrusion detection systems (IDS) an appropriate option. In this study, we propose an optimized model with high performance for intrusion detection in Message Queue Telemetry Transport protocol (MQTT)-based IoT networks for smart homes. This is done by studying 22 Machine Learning (ML) algorithms based on an extended two-stage evaluation approach that includes several aspects for optimizing and validating the performance to find the ideal model. Based on the empirical evaluation, the Generalized Linear Model (GLM) classifier with the random over-sampling technique produced the best detection performance with 100% accuracy and an f-score of 100%, outperforming previous studies. This study also investigated the influence of automatic feature engineering techniques on the performance of algorithms. With the automatic feature engineering technique, the performance increased by up to 38.9%, and the time required to classify the attacks decreased by up to 67.7%. This shows that automatic feature engineering can improve performance and reduce detection time.

INDEX TERMS IoT security, intrusion detection, machine learning, MQTT, smart homes, automatic feature engineering, resampling techniques.

I. INTRODUCTION

SMART homes have become popular in the past few years owing to their convenience and controllability. According to recent statistics, there are 360.72 million smart homes exist worldwide, and the number of smart homes is expected to increase by 86.74% by 2027 [1]. Smart home devices are typically connected to a central smart home hub (or ''gateway'') to provide the user with complete control and monitoring of the smart home either through a unified central system or through apps that come with each device [2]. Smart homes are a specific branch of the Internet of Things (IoT) that focuses on home appliances and uses IoT technologies to enable their functions.

The associate editor coordinating the review of this manuscript and approving it for publication was Sedat Akleylek.

The security aspect of IoT devices for smart homes is currently one of the most pressing concerns, especially given the expectation of a continued increase in the adoption of smart homes. A survey conducted from 2018 to 2022 showed that the global smart home security market has reached $4.3 billion in revenue [3]. However, there is a gap between the security requirements and capabilities of current IoT devices. One of the major challenges for IoT and related technologies is the detection of threats and vulnerabilities [4]. Most security issues in smart homes have been derived from the limitations of IoT hardware and the lack of security considerations when implementing and designing IoT software and protocols, such as MQTT [5]. Message Queue Telemetry Transport (MQTT) is the most widely used protocol in IoT, which is designed based on the need for a lightweight protocol that is compatible with resource-constrained and heterogeneous devices [6]. MQTT utilizes a publisher/subscriber model to enable lightweight messaging between devices [6].

In addition, the heterogeneity of IoT devices in terms of hardware, software, and protocols makes the implementation of a security mechanism that can handle the diversity of devices challenging. Moreover, IoT devices connected to the Internet are vulnerable to many intrusion attempts, and are exposed to security and privacy issues.

One of the security issues faced by the IoT is network intrusion, which can compromise security and privacy by accessing personal information, causing financial loss and eavesdropping [7]. To address this issue, Intrusion detection systems (IDSs) are used. There are many different proposed high-performance artificial intelligence (AI) models, making it difficult for developers to choose the best model. In addition, some aspects have not been considered when de- veloping models in previous studies, such as considering the MQTT protocol, solving the problem of data imbalance, and feature engineering. Further investigations are required to determine the best classification model for detecting MQTT attacks while considering all aspects of enhancement.

In this paper, we propose a Machine Learning (ML) model to effectively detect network intrusion in smart home IoT networks with improved performance. There are many factors that affect model performance such as machine learning algorithms, resampling techniques, feature engineering, hyperparameter tuning, and cross-validation techniques. Improving these factors helps to find the ideal model capable of detecting intrusion effectively.

We adopted an extended two-stage approach that addresses all factors that affect model performance. The first stage of this approach aims to analyze the performance of seven popular ML algorithms to select the three best classifiers for use in the second stage. The seven classifiers are Naïve Bayes (NB), Generalized Linear Model (GLM), Logistic Regression (LR), Fast Large Margin (FLM), Decision Tree (DT), Random Forest (RF), and Gradient Boost- ing Trees (GBT). The second phase aims to analyze five resampling techniques with each of the three classifiers selected during the first stage. Finally, with this approach, we were able to choose the best combination of classifiers and resampling techniques to propose the best model based on the best overall performance.

The significance of this study lies in proposing an efficient detection model to help developers choose the optimal solution for protecting smart homes from MQTT attacks. It can also help academic researchers follow the extended two-stage approach in various fields, including image and video recognition, identifying spam, detecting fraudulent transactions, etc. The model proposed in this paper can be applied as an IDS placed inside a smart home network in an intermediate point through which all traffic passes such as a router or broker (depending on the applied MQTT architecture).

The main contributions of this work can be summarized as follows:

- This paper presents an efficient detection model help developers to choose an optimal solution for protecting smart homes from MQTT attacks. The proposed model demonstrates improved intrusion detection performance to protect smart home devices from MQTT attacks in comparison with previous research.
- This study used an extended two-stage approach to produce models with improved performance, which can be utilized in different fields with different ML applications.
- This study is one of the first to apply automatic feature engineering to automatically extract a list of features that effectively contribute to intrusion detection and meet the requirements of IoT networks. The study found that automatic feature engineering significantly enhances the performance of the NB algorithm, out- performing the previous study [8] that used the same dataset, thereby reducing the time spent and enhancing performance.
- In comparison to previous research such as [8], [9], [10], and [11], which dealt with the problem of uneven distribution of classes, the performance of our proposed GLM- RandomOverSampling model outperforms in all as- pects, with accuracy, f-score, precision, and recall being equal to 100%

The rest of this paper is organized as follows: Section II presents previous related work. Section III describes the proposed classifier model. In Section IV, the results and a discussion are presented. Finally, the conclusions and suggestions for future work are presented in Section V.

II. RELATED WORK

Several studies have attempted to design IDS systems specialized for IoT networks. Researchers have applied ML algorithms for intrusion detection in IoT networks with promising results. For instance, Liu et al. [12] proposed an ML model using the DT classifier, which showed an accuracy of 100% after training and testing with three other classifiers. However, because of the small dataset containing only 480 records, the results seemed to be biased, showing high accuracy, as only 95 records were used to test the model.

Additionally, the authors of [8] propose a three-layer IDS architecture that determines whether an attack occurred, where it occurred, and what type of attack it was. The J48 classifier showed the best performance, with an F-measure of 88.8% for attack detection on unseen data.

Most researchers employing ML for intrusion detection in IoT networks have tended to use datasets generated from IT networks that contain traditional attacks and apply them in the context of IoT networks. The traffic of IoT networks includes protocols that are different from the protocols of the traditional IT network, which makes its features different. The authors in [7] used the CIDDS-001, UNSW-NB15, and NSL-KDD datasets to protect IoT networks from DoS attacks. In terms of high accuracy, low false positive rate (FPR), and fast response time, researchers found that the Classification and Regression Algorithm (CART) showed the best trade-off with an accuracy of 96.74%.

Another study [12] used the NSL-KDD dataset to inspect DoS and probe attacks in routing protocols that can affect sensor nodes in IoT networks. The authors concluded that the DT classifier is suitable for use in a sink node in IoT networks with an accuracy of 96.6% and an F-measure of 96.8%. The datasets used in the studies mentioned earlier were primarily created for IT networks, rather than IoT networks. Even researchers who have built their own IoT network environment lack consideration of IoT protocol attacks when generating datasets.

More effort is still needed to consider the detection of attacks that occur in IoT protocols, such as the MQTT protocol. Over the past few years, researchers have attempted to address this gap by considering the MQTT protocol when proposing ML models. One study [13] built an MQTT-based network and created a dataset that included normal traffic, generic network attacks, and MQTT-based attacks. The dataset was published for use in the scientific community as the MQTT-IoT-IDS2020 dataset. In their work, the RF classifier gave the highest F-measure of 99.97% for flow-based features, and the DT classifier achieved the highest F-measure of 88.54% for packet-based features. In addition, Alatram et al. [14] proposed a dataset called DoS/DDoS-MQTT-IoT to evaluate intrusion detection in IoT networks that use the MQTT protocol, filling a gap in this domain. A comparison of classifiers was conducted with all the extracted 30 different features, based on which two algorithms were suggested: RF and XGBoost with accuracies of 92.69%, and 92.72% respectively. The dataset contains normal and attack traffic for ten types of DoS attacks.

Another study [15] proposed the implementation of an IDS that uses a Principal Component Analysis (PCA) classifier trained using a dataset created from an MQTT- based IoT network. PCA revealed the best Area Under the Curve (AUC) of $> 89\%$ and the shortest training time. In addition, a previous study [16] developed an IDS prototype that detects anomalies in MQTT-based IoT environments. After training and testing the DT classifier and k-nearest neighbors algorithm (KNN), the results showed that the DT's performance outperformed the other with an average F-measure of 93.81%. However, the imbalanced dataset issue was not addressed or resolved in any of the three previous studies [13], [15], [16].

In classifying real-world problems, the available datasets are often unbalanced, containing too much data for normal behavior (majority class) and too little data for abnormal behavior (minority class). The problem with this imbalance is that the classification algorithm can be biased towards the majority class, simply because they do not have sufficient data to identify the minority class [17]. The minority class is the most important class for detection in the classification process, especially for detecting intrusions in IoT networks. The success of detection is highly influenced by the selection of appropriate features that balance accuracy and complexity. It also helps avoid the impact of large amounts of data generated by the IoT environment [18]. Studies involving intrusion detection in MQTT-based IoT networks lack focus on feature engineering, and only a few have considered this point. For instance, the authors of [9] created and published an MQTT-based dataset named "MQTTset" with many features. In their work, they emphasized the im- portance of extracting the most influential features from raw data to improve the accuracy of the classifier; nevertheless, the method of feature extraction was not discussed, and automatic feature engineering was not used. The imbalance issue was solved using the random over-sampling technique. They tested five classifiers, and the RF classifier yielded the best results, with an F-measure of 91.40%. In [10], a TNN-IDS was proposed for MQTT-enabled IoT networks, and its performance was evaluated using MQTT-IoT-IDS2020. According to the results, the Uni-Flow, Bi-Flow, and Packet-Flow features were able to detect malicious activity with the best F-measure of 99.99%, 99.97%, and 99.97%, respectively, for the transformer neural network (TNN). Feature filtering was applied as a selection method.

Moreover, Alaiz-Moreton et al. [11] chose the most representative features using a feature importance (FIM) report system. They proposed several models, including the ensemble method XGBoost and deep learning (DL) classifiers long short-term memory (LSTM) and gated recurrent unit (GRU), which achieved an F-measure of 95.78%. However, they did not present the selected features, and the resampling technique used to address the imbalance problem was not stated.

The authors of [19] selected four features from the MQTT-IoT-IDS2020 dataset using the RF algorithm as a feature selection method. The authors tested seven classifiers, and the RF classifier outperformed them all, with an F- measure of 99.62%. Similarly, another study [20] presented seven features selected using correlation-based network fea- ture selection. They presented a fuzzy logic-based intrusion detection model designed for protecting IoT nodes using the MQTT protocol against DoS attacks. However, neither study has resolved the issue of an imbalanced dataset.

A summary of previous related works is presented in Table 1. Generally, most of the previously mentioned studies lack one or more of the following aspects: either models are not being evaluated using IoT-specific datasets, they do not handle the imbalanced data issue, or they lack focus on fea- ture engineering. Most studies that use unbalanced datasets provide high results. This is because when the dataset is unbalanced the accuracy can be high even though the model fails to recognize the minority class. Consequently, the accuracy of the model in determining both classes is questionable, which prompted us to investigate further and consider balancing the dataset to obtain a reliable model with an accurate performance.

TABLE 1. Summary of related work.

Paper Ref.	Year	Classifier	Performance	Dataset	MQTT Protocol	Handle data imbalance issue	Automatic Feature Eng.
[21]	2020	DT	F-score = 100%	Generated dataset	No	No	No
[8]	2019	J48	F-score = 88.8%	Generated dataset	No	Yes (Random under sampling)	No
[7]	2020	CART	ACC= 96.74%	CIDDS-001 UNSWNB15 NSL-KDD	No	No	No
[12]	2020	XGBoost	F-Score = 96.8%	Generated dataset using attacks in NSL-KDD	No	No	No
[13]	2021	DT	Avg. F-score= 88.54% for Packet-based features	Generated dataset (MQTT-IoT-IDS2020)	Yes	No	No
[14]	2023	RF and XG-Boost	ACC= 92.69%, and 92.72% respectively	Generated dataset (DoS/DDoS-MQTT-IoT)	Yes	No	No
[15]	2022	PCA	AUC= 89.296%	Generated dataset	Yes	No	No
[16]	2021	DT	Avg. F-score= 93.81%	Generated dataset	Yes	No	No
[9]	2020	RF	F-score = 91.40%	Generated dataset (MQTTset)	Yes	Yes (Random Over-sampling)	No
[10]	2023	TNN	Avg. F-score= 99.97% for Packet-based features	MQTT-IoT-IDS2020	Yes	Yes (Not specified)	No (Feature Filtering approach- ETC Selection)
[11]	2019	XGBoost LSTM GRU	F-score= 93.28% F-score= 95.78%	Generated dataset	Yes	Yes (Not specified)	No (FIM)
[19]	2022	RF	F-score= 99.62%	MQTT-IoT-IDS2020	Yes	No	No (RF)
[20]	2019	Fuzzy logic	F-score = 90.90%	Generated dataset	Yes	No	No (Correlation-based)

To overcome these limitations, we used our extended two-stage approach, which includes several techniques to improve the performance of classifiers and thus find the best model to achieve our goal. The key objective of this project is to develop an ML model that can effectively detect malicious behavior in MQTT-based IoT networks for smart homes. The automatic feature engineering method used in our study is the first among ML-based IDS studies in MQTT-based IoT networks.

III. SYSTEM DESIGN AND IMPLEMENTATION

A. EXTENDED TWO-STAGES APPROACH

In this study, we propose an improved intrusion detection model for MQTT-based IoT networks in smart homes. To develop the model, we follow a two-stage approach to select the best classifier and sampling technique. A schematic of the workflow of our study is shown in Fig. 1, where the extended approach is divided into two stages, and the results of the first stage are used as inputs to the second stage, as inspired by the approach in [22]. The steps of the extended two-stage approach are illustrated in Fig. 2.

B. EXPERIMENTS ENVIRONMENT

All experiments were conducted on a 64-bit Windows 10 laptop equipped with an Intel Core i7-7700HQ 2.80GHz processor and 16GB RAM. The experiment was performed using Anaconda Navigator 2.3.2, Jupyter Notebook 6.5.2, Python 3.10.9, and RapidMiner Studio 9.10.011. For the proposed model, the GLM algorithm was implemented using H2O 3.30.0.1. The parameters used for this algorithm are as follows: family = binomial, solver = auto, link= logit, the maximum number of threads = 1, regularization (it will make H2O calculates values for lambda, alpha, and the lambda search-related parameters based on the training data and the other parameters), standardize, add_intercept, max_iterations = 0.

C. DATASET DESCRIPTION

We conducted this study using the MQTT-IoT-IDS2020 dataset. This is a recent dataset published by [13] in 2020, which was the first dataset generated from a real MQTT-based IoT network. This dataset contains recorded traffic for normal operation and four attack scenarios: aggressive scan, UDP scan, Sparta SSH brute-force attack, and MQTT brute- force attack. The dataset is divided into three abstraction levels of features: packet-based, unidirectional-based, and bidirectional-based. In our research, we used the packet- based features dataset, as it contains the packet MQTT fields we need to build our model, which contains the most comprehensive set of features among the three. Each

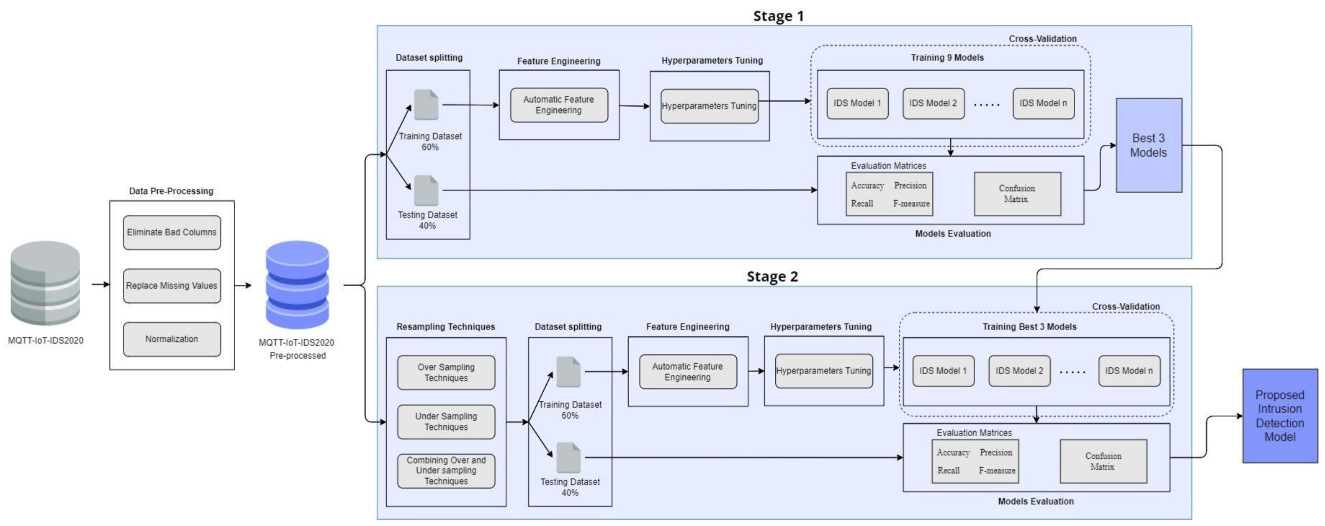


FIGURE 1. A schematic view of our study workflow.

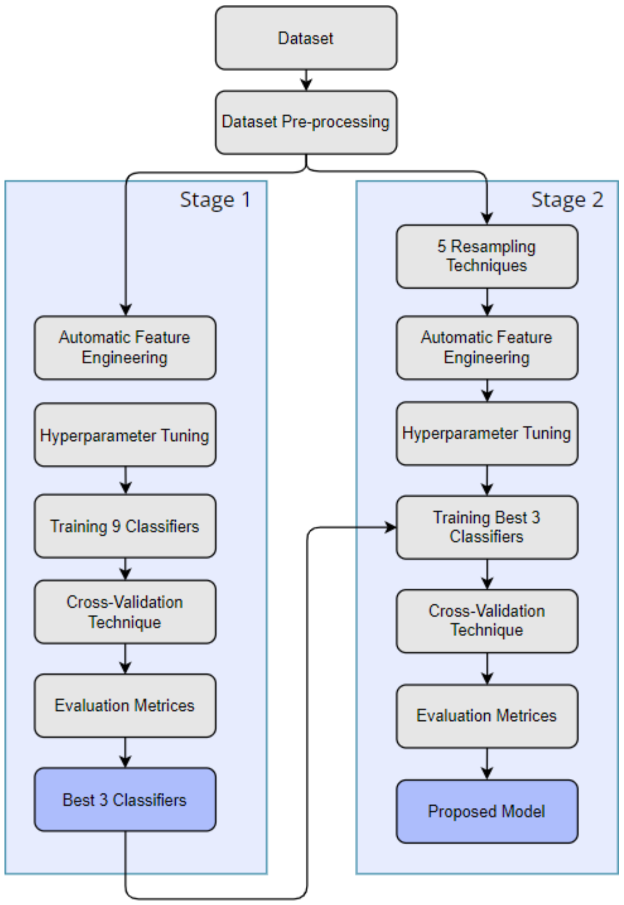


FIGURE 2. The extended two-stages approach of the study.

entry was processed as an attack or normal. The distribution of the dataset records used in this study is presented in Table 2.

D. DATA PREPROCESSING

To ensure that the model does not depend on specific network configurations and network devices, we removed some columns that could affect the model performance, including source IP addresses, destination IP addresses, and timestamps. Subsequently, we replaced the missing values in the dataset. Finally, we normalized the numerical columns to ensure they were on the same scale.

Before moving to the first stage, we split the dataset into two sets: one with 60% of the data that was used in training and the other with 40% of the data that was unseen to the model and used for testing.

E. AUTOMATIC FEATURE ENGINEERING

To choose the most influential feature, we utilized an automatic feature engineering technique to improve the model performance by employing the power of the most relevant features [23]. Automatic feature engineering is the process of automatically selecting or building new features from existing ones [24], which is a trade-off between feature complexity and model performance. It helps reduce the feature space which helps decrease the training time with improved performance. Coming up with features is difficult, time-consuming, and requires expert knowledge; therefore, automated feature engineering improves the traditional ap- proach to feature engineering with a framework that can be applied to any problem. This method has not been investigated in previous studies on MQTT related attacks; therefore, in our study, we trained the model with and without automatic feature engineering to determine its effect on performance.

F. HYPERPARAMETER TUNING

Hyperparameter tuning is the process of choosing a set of optimal hyperparameters for a learning algorithm that maximizes model performance and minimizes errors to produce

TABLE 2. Dataset distribution.

	Cyber Attack Types							
	Aggressive scan		UDP scan		MQTT Brute-force		Sparta SSH brute force	
	Normal	Attack	Normal	Attack	Normal	Attack	Normal	Attack
Records	70,768	40,624	210,819	22,436	1,281	398,719	13,743	286,257
	Total		Total		Total		Total	
	111,392		233,255		400,000		300,000	

TABLE 3. Five resampling techniques.

Category	Category
Under-sampling	Random Under-sampling Near Miss v1
Over-sampling	Random Over-sampling SMOTE
Both Under-sampling and Over-sampling	SMOTETomek

better results [25]. Models may have several hyperparameters and determining the best combination of parameters can be treated as a search problem. The method used for tuning the hyperparameters was the grid search implementation in RapidMiner.

G. MACHINE LEARNING ALGORITHMS

The classification process refers to the act of predicting a class label for a given unlabeled point [26]. Seven well-known classifiers have been utilized to model the data:

1) Naive bayes (NB): This is a probabilistic classifier built on Bayes' theorem with the assumption of fully independent features [26].
2) Generalized linear model (GLM): This is a generalization of conventional linear models that utilize a link function to connect the linear model to the response variable and make the variance of each measurement a function of its predicted value [27].
3) Logistic regression (LR): Is a type of statistical model that models the probability of a data point belonging to a specific class based on independent data points, and assumed the points are distributed according to a linear function [26].
4) Fast Large Margin (FLM): Applies a fast margin learner based on the linear SVM scheme proposed by [28]. It offers results similar to classical SVM or logistic regression, but it can handle data with millions of examples and attributes.
5) Decision tree (DT): It predicts the class for each point based on the recursive partition-based tree model. It recursively partitions the data space into "pure" regions that contain data points from one class with relatively few exceptions [26].
6) Random forest (RF): Is an ensemble learning that operates by constructing a collection of decision trees "forest" at training time [26]. Random forests eliminate the limitations of the decision trees. It reduces the overfitting of datasets and increases accuracy [7].
7) Gradient boosting tree (GBT): Is an ensemble learning model that builds a series of trees in a sequential manner, and can be used for regression and classification [29]. Each tree relies on the predictions of the previous tree to improve prediction errors.

H. CROSS VALIDATION TECHNIQUE

Cross-validation is a technique for validating the effectiveness of a model and testing its performance by training it on a subset of the dataset and testing it on another unseen subset [30]. The accurate validation of a model may not be possible with a single evaluation iteration. To ensure that the model undergoes rigorous evaluation and achieves robust performance, the repeated iteration approach is the best option. In our research, we performed seven iterations of cross-validation, which is called 7-fold cross-validation [30]. The average value of all seven iterations is considered to be the final value when evaluating the model using the evaluation metrics.

I. EVALUATION METRICS

To assess the performance of the trained classifiers, the following measures were calculated for each model in both stages: accuracy, precision, recall, F-measure, and false alarm rate (FAR), along with measurements of the time spent for training and testing in milliseconds. These measurements depend on the confusion matrix. A confusion matrix is a table used to define the performance of the classification algorithm, which provides four outcomes: True Positive (TP), False Negative (FN), True Negative (TN), and False Positive (FP). A scheme of the confusion matrix for binary classification is shown in Fig. 3.

The percentage of correctly identified attacks is called the accuracy. Equation (1) shows the Accuracy calculated as the ratio of correctly classified cases (attack (TP) and normal (TN)) to the total number of cases.

$$Accuraccy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

Precision, also called the Positive Predictive Value, represents the percentage of accurately detected attacks out of the total number of traffic labeled as an attack. Equation (2) shows the formula used to calculate the precision:

$$Percision = \frac{TP}{TP + FP} \tag{2}$$

Recall, also known as the Sensitivity and True Positive Rate (TPR), is the percentage of attack traffic, appropriately

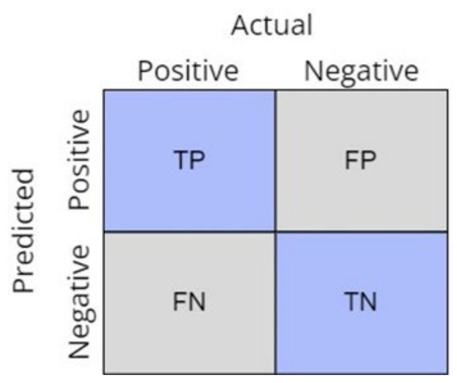


FIGURE 3. Confusion matrix.

labeled out of all attack traffic as shown in Equation (3):

$$Recall = TPR = \frac{TP}{TP + FN} \tag{3}$$

The F-score is a combined measure of precision and recall values as shown inEquation (4).

$$F\_Score = 2\frac{Percision \times Recall}{Percision + Recall} \tag{4}$$

The False Alarm Rate (FAR), also known as the False Positive Rate, is the percentage of negative events incorrectly classified as positive (false positives) compared to the total number of negative events. Equation (5) shows the formula used to calculate the FAR:

$$FAR = \frac{FP}{FP + TN} \tag{5}$$

The accuracy measure is ineffective in evaluating models on an unbalanced dataset, whereas the F-score is a better metric when there are imbalanced classes [31]. Thus, the F- score can be considered more reliable than accuracy when comparing two or more models.

J. RESAMPLING TECHNIQUE

Resampling techniques are typically used to address the imbalance class issue in a dataset [32]. The uneven balance of classes in the MQTT-IoT-IDS2020 dataset could negatively impact classification performance. When the dataset is imbalanced, the classification algorithm can be biased towards the majority class, simply because there is insufficient data to identify minorities. There are three main categories of resampling techniques: under-sampling, over-sampling, and a combination of both under-sampling and over-sampling.

The under-sampling technique is used to balance uneven datasets by keeping all of the data in the minority class and decreasing the size of the majority class. This reduces the training time, but the problem is that it removes a large portion of the training set, which leads to the loss of valuable information that would lead to difficulties in classification and prediction [33].

In contrast to under-sampling, over-sampling aims to preserve the majority class instances while replicating mi- norities. The issue with this technique is that overfitting can occur and it may be difficult to generate minority data in the training set which could lead to poor performance [22], [34].

In the last technique, both under-sampling and over- sampling are combined at the same time. By merging these techniques, the imbalance class issue can be addressed differently. We balanced the dataset using the resampling methods provided in the Scikit-learn library [35]. Table 3 lists the five popular resampling techniques selected in this study.

K. THE PROPOSED MODEL

In this study we propose an improved intrusion detection model for MQTT-based IoT networks in smart homes. The problem is formulated as a binary classification problem, in which we distinguish between two classes: normal and malicious. Based on the influential features and the best performing classifier, we propose an intrusion detection model. Fig. 4 shows the architecture of the proposed model. The classification process starts by inputting the MQTT-based IoT raw traffic into the IDS architecture. Then, the IDS extracts the relevant features, the traffic is preprocessed to be normalized, and the missing values are handled, and finally, the traffic is fed to the trained GLM-RandomOverSampling model that has been selected based on the results discussed in the next section. It then classifies the type of traffic and detects attacks, if any.

IV. RESULTS

A. THE FIRST STAGE: ALGORITHM COMPARISON

In the first stage, we investigated the influence of automatic feature engineering on the performance of the algorithms. We ran the experiment once without using automatic feature engineering and again with it to observe the difference in the results. Table 4 shows the average results for detecting all types of attacks after applying the first experiment without the use of automatic feature engineering, whereas Table 5 presents the results with the use of automatic feature engineering.

The results show that the use of automatic feature engineering increases performance and reduces the time required for training and testing. In terms of accuracy, the GBT increased by 38.9% from 71.5% to 99.3%. Although the accuracy does not reflect the overall performance in unbalanced data, the recall confirms this apparent improvement in accuracy, as its value increased from 62.5% to 99.9%. NB showed the best improvement in precision, increasing from 86.5% to 100.0%. In terms of F-score, GBT also showed the best improvement, increasing from 88.8% to 99.6%.

Regarding the time consumed, the DT algorithm shows the best test time improvement of 67.7% compared with without using automatic feature engineering. For the FLM, the training time was reduced by 92.5%. Therefore, au- tomatic feature engineering effectively reduces the time required and improves the performance of AI models. How- ever, the

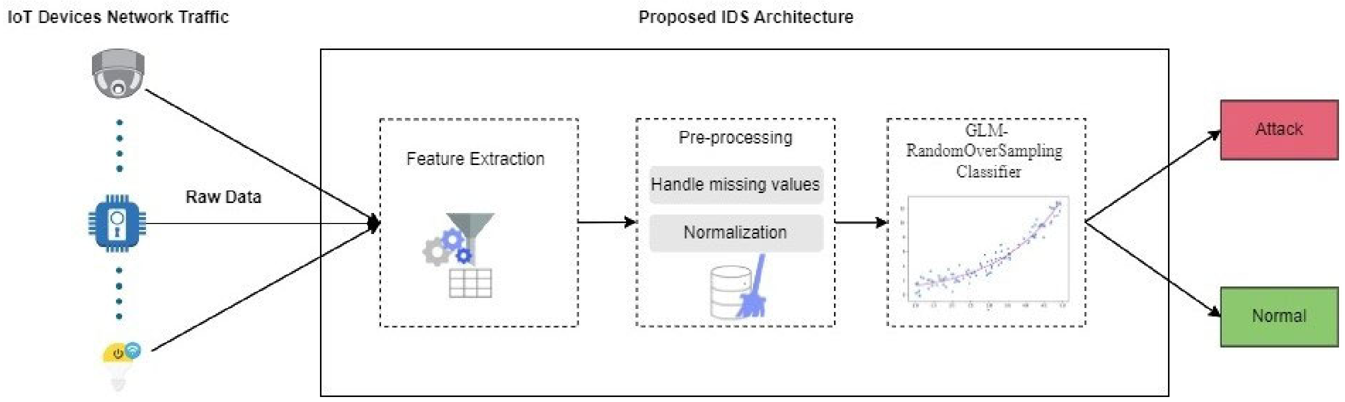


FIGURE 4. The architecture of the proposed model.

TABLE 4. Average results of stage one without using automatic feature engineering.

Classifier	Accuracy	F-score	Precision	Recall	FAR	Training Time (ms)	Testing Time (ms)
NB	89.7%	91.0%	86.5%	**100.0%**	17.04%	4.5	277.8
GLM	99.0%	99.3%	98.6%	99.3%	3.31%	23.3	216.0
LR	99.1%	99.5%	99.1%	**100.0%**	2.83%	23.7	242.3
FLM	**99.6%**	**99.8%**	99.6%	**100.0%**	1.50%	70.3	**198.0**
DT	90.2%	99.5%	99.0%	75.0%	2.49%	**3.5**	203.3
RF	89.9%	88.4%	**100.0%**	81.7%	0.04%	21.8	502.3
GBT	71.5%	88.8%	99.9%	62.5%	0.20%	39.3	237.5
SVM	87.5%	91.7%	**100.0%**	87.5%	**0.03%**	297.8	425.0

TABLE 5. Average results of stage one.

Classifier	Accuracy	F-score	Precision	Recall	FAR	Training Time (ms)	Testing Time (ms)
NB	**100.0%**	**100.0%**	**100.0%**	99.9%	7.16%	2.5	104.0
GLM	99.7%	99.7%	99.3%	**100.0%**	0.45%	11.3	121.8
LR	99.3%	99.7%	99.3%	**100.0%**	2.90%	17.5	141.8
FLM	**100.0%**	**100.0%**	**100.0%**	**100.0%**	0.04%	5.3	129.3
DT	98.8%	99.1%	98.3%	99.9%	3.43%	**2.0**	**65.8**
RF	95.4%	91.6%	**100.0%**	87.4%	**0.03%**	7.5	206.8
GBT	99.3%	99.6%	99.3%	99.9%	1.79%	18.8	96.5
SVM	79.8%	82.0%	98.4%	79.2%	0.71%	555.5	400.0

results shown at this stage do not reflect the actual performance of these algorithms, given the data imbalance and model bias towards the dominant class.

The goal of the first stage was an initial assessment of the performance of each ML algorithm. According to Table 5, the FLM algorithm shows the best overall performance of up to 100% for all the evaluation metrics. Followed by the NB algorithm with accuracy, F-score, precision up to 100%, and recall with 99.9%. This was followed by GLM, with accuracy and F-score of 99.7% and precision of 99.3% and recall of 100%. Thus, FLM, NB, and GLM are the three ML algorithms selected in the first stage.

B. THE SECOND STAGE: RESAMPLING TECHNIQUES

In the second stage, we combined the three best selected classifiers with five resampling techniques to solve the data imbalance issue. Tables 6, 7 and 8 show the average results of the second stage, where each table represents one of the selected classifiers with each resampling techniques. As shown in Table 8, the GLM showed the highest accuracy, but we could not determine the best model based on accuracy alone. In terms of F-score, NB, and GLM show the highest value with the random over-sampling technique as shown in Tables 7 and 8, which is a good indication of this resampling technique. The F-score is an extremely valuable evaluation metric that provide an important indication of the best model so far [22].

Another important evaluation metric is precision, which indicates the percentage of appropriately classified positive cases out of all the positive cases. NB and GLM also showed the highest precision with the random over-sampling technique. Regarding recall, many classifiers with resampling techniques have achieved similarly high values. But, when the recall value is too low and the precision value is too high, this indicates that the model has high assumptions of classifying positive cases [22]. This means that there are

TABLE 6. Fast large margin with resampling techniques.

Resampling	Accuracy	F-score	Precision	Recall	FAR	Training Time (ms)	Testing Time (ms)
Random Under sampling	94.6%	94.5%	94.5%	94.7%	2.23%	51.8	158.5
Near Miss v1	**99.0%**	**99.0%**	98.1%	**100.0%**	**0.13%**	64.5	183.3
Random Over Sampling	98.8%	98.9%	98.0%	99.9%	2.83%	61.0	112.8
SMOTE	98.7%	98.6%	**99.9%**	97.5%	0.20%	**6.0**	**68.3**
SMOTETomek	94.9%	94.8%	94.9%	94.8%	6.40%	43.5	143.3

TABLE 7. Naive Bayesian with resampling techniques.

Resampling	Accuracy	F-score	Precision	Recall	FAR	Training Time (ms)	Testing Time (ms)
Random Under sampling	96.5%	96.1%	98.9%	94.1%	2.46%	13.8	179.5
Near Miss v1	98.1%	98.2%	96.6%	99.8%	2.93%	43.8	300.0
Random Over Sampling	87.5%	**100.0%**	**100.0%**	75.0%	**0.00%**	**2.3**	**85.5**
SMOTE	87.7%	91.3%	87.0%	99.5%	8.31%	2.8	112.5
SMOTETomek	**99.0%**	99.0%	98.1%	**100.0%**	3.40%	2.8	124.0

TABLE 8. Generalized linear model with resampling techniques.

Resampling	Accuracy	F-score	Precision	Recall	FAR	Training Time (ms)	Testing Time (ms)
Random Under sampling	93.4%	92.4%	99.9%	86.9%	0.26%	29.0	202.8
Near Miss v1	99.7%	99.7%	**100.0%**	99.5%	**0.00%**	33.5	142.8
Random Over Sampling	**100.0%**	**100.0%**	**100.0%**	**100.0%**	**0.00%**	9.5	113.8
SMOTE	98.7%	98.6%	99.9%	97.5%	0.19%	**8.0**	**86.8**
SMOTETomek	99.0%	99.0%	98.1%	**100.0%**	3.38%	9.3	134.8

many more positive cases stated than the actual number of positive cases. This can be seen with the NB model using random over-sampling in Table 7, which gives a low recall value of 75%, but a high precision value of 100%. Also, seen in the FLM model with Near Miss v1 undersampling in Table 6. Therefore, the results achieved by the NB with the random over-sampling technique should not be considered to reflect good performance.

Based on the stage 2 results, the GLM with random over-sampling achieves the highest results with 100% on all evaluation matrices and takes a relatively short testing time and FAR value of 0.26%. Table 9 shows the set of features extracted from the automatic feature engineering, which helped the GLM algorithm detect attacks.

Fig. 5 shows the confusion matrix for the GLM with random over-sampling for detecting each type of attack in the dataset. The calculated performance from the confusion matrix can differ slightly from those shown in the performance table. It is because the performance tables were calculated after applying the cross-validation technique to the unseen test sample (40% of the dataset). In which was divided into seven-folds, each having different performance values, and then the outlier performances were removed, and finally the average performance over the remaining five performances was calculated.

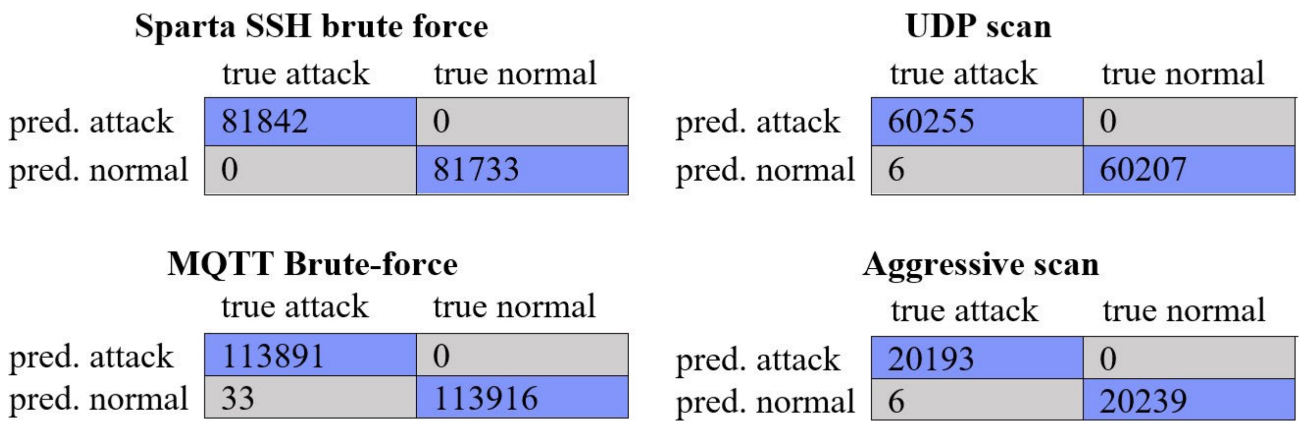
By analyzing the confusion matrices, the GLM correctly classifies Sparta SSH brute force traffic without mistaking any of the classes. While 6 packets have been mistakenly classified as normal traffic in the UDP scan and Aggres- sive scan attacks. Also, there are 33 Brute-force MQTT traffic packets incorrectly classified as legitimate traffic. Since some of the MQTT-based attacks depend on available MQTT normal commands (such as publish, subscribe, and etc.), it is difficult to distinguish attacks from legitimate operations where the same commands are used.

C. DISCUSSION

Our goal is to propose an efficient ML based IDS model to detect MQTT attacks; so we investigated the effect of using automatic feature engineering on the performance as none of the previous works used automatic feature engineering. The accuracy of the NB algorithm increases from 89.7% to 100% when using automatic feature engineering. This result outperformed those of previous studies [8], [9], [10], [12], [13], [21] when using automatic feature engineering in the NB algorithm. Although the studies of [13] and [10] used the same dataset, we achieved better performance. We obtained an accuracy, f-score, and precision of 100% and recall of 99.9% for the NB algorithm, which outperformed the results [13] obtained as an accuracy, f-score, precision,

TABLE 9. Featureset extracted by automatic feature engineering.

Feature set	Description	Attack
mqtt_messagetype	MQTT message type	Aggressive scan / UDP scan
mqtt_flag_uname	Username MQTT Flag	Aggressive scan / UDP scan
mqtt_flag_passwd	Password MQTT flag	Aggressive scan / UDP scan
mqtt_flag_retain	Will retain MQTT flag	Aggressive scan / UDP scan
mqtt_flag_qos	Will QoS MQTT flag	Aggressive scan / UDP scan
mqtt_flag_willflag	Will flag MQTT flag	Aggressive scan / UDP scan
mqtt_flag_clean	Clean MQTT flag	Aggressive scan / UDP scan
mqtt_flag_reserved	Reserved MQTT flag	Aggressive scan / UDP scan
Protocol	Last layer protocol	Aggressive scan / UDP scan
src_port	Source Port	Aggressive scan / UDP scan
dst_port	Destination Port	MQTT Brute-force / Aggressive scan / UDP scan
ttl	Time to live	Aggressive scan / UDP scan
ip_len	Packet Length	Aggressive scan / UDP scan / Sparta SSH brute force
ip_flag_df	Don't fragment IP flag	Aggressive scan / UDP scan
ip_flag_mf	More fragments IP flag	Aggressive scan / UDP scan
ip_flag_rb	Reserved IP flag	Aggressive scan / UDP scan
tcp_flag_cwr	Congestion Window Reduced TCP flag	MQTT Brute-force / Aggressive scan / UDP scan
tcp_flag_reset	Reset TCP flag	MQTT Brute-force / Aggressive scan / UDP scan / Sparta SSH brute force
tcp_flag_ns	Nonce sum TCP flag	MQTT Brute-force / Aggressive scan / UDP scan
tcp_flag_urg	Urgent TCP flag	MQTT Brute-force / Aggressive scan / UDP scan
tcp_flag_res	Reserved TCP flag	Aggressive scan / UDP scan
tcp_flag_ecn	ECN Echo TCP flag	Aggressive scan / UDP scan
tcp_flag_ack	Acknowledgement TCP flag	Aggressive scan / UDP scan
tcp_flag_push	Push TCP flag	Aggressive scan / UDP scan
tcp_flag_syn	Synchronization TCP flag	Aggressive scan / UDP scan
tcp_flag_fin	Finish TCP flag	Aggressive scan / UDP scan



Sparta SSH brute force

	true attack	true normal
pred. attack	81842	0
pred. normal	0	81733

UDP scan

	true attack	true normal
pred. attack	60255	0
pred. normal	6	60207

MQTT Brute-force

	true attack	true normal
pred. attack	113891	0
pred. normal	33	113916

Aggressive scan

	true attack	true normal
pred. attack	20193	0
pred. normal	6	20239

FIGURE 5. Confusion matrix of GLM in detecting each attack.

and recall equal to 81.15%, 75.99%, 73.29%, and 81.15%, respectively. Therefore, we conclude that automatic feature engineering enhances the performance of the ML algorithm and reduces the time required.

Few previous studies have addressed the issue of class imbalance. After analyzing the results, we noticed that the performance decreased in most classifiers after addressing the imbalance issue, which is consistent with the findings of [9]. This is because the models were biased towards the majority class before handling the data imbalance, unaffected by incorrect classification of minority classes. However, after addressing the unbalanced distribution of the classes, the models classified both fairly, decreasing performance and making them more realistic.

Compared to previous studies [8], [9], [10], [11] that dealt with the problem of uneven distribution of classes, the performance of our proposed GLM-RandomOverSampling model outper- forms it in all aspects, with accuracy, f-score, precision, and recall being equal to 100%. Although the study of [9] addresses the problem of data imbalance using the same resampling technique (random over-sampling), we obtained a performance equivalent to 100% and a test time of 113.8 ms, while in their study they obtained an accuracy of 91.59%, an f-score of 91.40%, and a test time of 144.2 s for their proposed model. Therefore, our proposed GLM- RandomOverSampling model, after solving the class imbal- ance problem, performed better than the previous studies men- tioned in this paper.

V. CONCLUSION

With the increasing popularity of smart homes, it has become crucial to protect them from cyber-attacks that breach the privacy of home residents. The MQTT protocol is one of the protocols used by IoT devices in smart homes. However, this protocol lacks strong security measures, which leaves MQTT communications open to security breaches. This study aims to improve the protection of MQTT-based IoT networks by utilizing ML algorithms.

Researchers have explored various ML algorithms for detecting IoT attacks with high accuracy rates. However, developers struggle to determine the optimal solution due to different proposed models. Therefore, this study uses an extended two-stage approach to compare classifiers and select the best performing model. This approach considers performance enhancement factors, including automatic feature engineering, hyperparameter tuning, cross-validation technique, and resampling technique. This extended approach can be applied in various fields, including image and video recognition, identifying spam, detecting fraudulent transactions, etc.

This study proposes an efficient ML model for detecting intrusion in IoT networks of smart homes using the GLM-RandomOverSampling algorithm. The model achieves 100% accuracy, f-score, precision, and recall when trained and tested using the MQTT-IoT-IDS2020 dataset. Also, the study investigates the effect of automatic feature engineering on model performance, and found an increase in accuracy of 38.9% and a decrease in attack detection time of 67.7%. The extracted features can be used when developing an IDS system, reducing time and improving performance better than examining all traffic features.

For future work, we will compare the performance of the models using additional datasets with different attack types, as the current study is limited to one dataset. In addition, more ML algorithms such as (K-nearest neigh- bor (KNN), light gradient-boosting machine (LightGBM)), DL algorithms such as (Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN)), resampling techniques (such as Adaptive Synthetic (ADASYN), One- Sided Selection), feature engineering such as (Deep Feature Synthesis (DFS), SULOV with Recursive XGBoost), and other optimization algorithms such as (Elephant Herding Optimization (EHO), Slime Mold Algorithm (SMA)) will be considered. Finally, we are planning to apply multiclass classification to be able to identify the type of detected attacks.

REFERENCES

[1] J. Lasquety-Reyes. (2018). *Number of Users of Smart Homes Worldwide From 2018 To 2027.* [Online]. Available: https://www.statista.com/forecasts/887613/number-of-smart-homes-in-the-smart-home-market-in-the-world

[2] D. Mocrii, Y. Chen, and P. Musilek, "IoT-based smart homes: A review of system architecture, software, communications, privacy and security," *Internet Things*, vols. 1–2, pp. 81–98, Sep. 2018.

[3] B. Thormundsson. (Nov. 2022). *Smart Home Security Market Value Worldwide 2018–2026*. [Online]. Available: https://www.statista.com/statistics/1056057/worldwide-smart-home-security-market-value/

[4] F. Arat and S. Akleylek, "Attack path detection for IIoT enabled cyber physical systems: Revisited," *Comput. Secur.*, vol. 128, May 2023, Art. no. 103174, doi: 10.1016/j.cose.2023.103174.

[5] F. Arat and S. Akleylek, "A new method for vulnerability and risk assessment of IoT," *Comput. Netw.*, vol. 237, Dec. 2023, Art. no. 110046, doi: 10.1016/j.comnet.2023.110046.

[6] A. J. Hintaw, S. Manickam, M. F. Aboalmaaly, and S. Karuppayah, "MQTT vulnerabilities, attack vectors and solutions in the Internet of Things (IoT)," *IETE J. Res.*, vol. 69, no. 6, pp. 3368–3397, Aug. 2023.

[7] A. Verma and V. Ranga, "Machine learning based intrusion detection systems for IoT applications," *Wireless Pers. Commun.*, vol. 111, no. 4, pp. 2287–2310, Apr. 2020.

[8] E. Anthi, L. Williams, M. Slowinska, G. Theodorakopoulos, and P. Burnap, "A supervised intrusion detection system for smart home IoT devices," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 9042–9053, Oct. 2019.

[9] I. Vaccari, G. Chiola, M. Aiello, M. Mongelli, and E. Cambiaso, "MQTTset, a new dataset for machine learning techniques on MQTT," *Sensors*, vol. 20, no. 22, p. 6578, Nov. 2020.

[10] S. Ullah, J. Ahmad, M. A. Khan, M. S. Alshehri, W. Boulila, A. Koubaa, S. U. Jan, and M. M. Iqbal Ch, "TNN-IDS: Transformer neural network-based intrusion detection system for MQTT-enabled IoT networks," *Comput. Netw.*, vol. 237, Dec. 2023, Art. no. 110072, doi: 10.1016/j.comnet.2023.110072.

[11] H. Alaiz-Moreton, J. Aveleira-Mata, J. Ondicol-Garcia, A. L. Muñoz-Castañeda, I. García, and C. Benavides, "Multiclass classification procedure for detecting attacks on MQTT-IoT protocol," *Complexity*, vol. 2019, pp. 1–11, Apr. 2019.

[12] J. Liu, B. Kantarci, and C. Adams, "Machine learning-driven intrusion detection for Contiki-NG-based IoT networks exposed to NSL-KDD dataset," in *Proc. 2nd ACM Workshop On Wireless Secur. Mach. Learn.* New York, NY, USA: Association for Computing Machinery, Jul. 2020, pp. 25–30.

[13] H. Hindy, E. Bayne, M. Bures, R. Atkinson, C. Tachtatzis, and X. Bellekens, "Machine learning based IoT intrusion detection system: An MQTT case study (MQTT-IoT-IDS2020 dataset)," 2020, *arXiv:2006.15340*.

[14] A. Alatram, L. F. Sikos, M. Johnstone, P. Szewczyk, and J. J. Kang, "DoS/DDoS-MQTT-IoT: A dataset for evaluating intrusions in IoT networks using the MQTT protocol," *Comput. Netw.*, vol. 231, Jul. 2023, Art. no. 109809, doi: 10.1016/j.comnet.2023.109809.

[15] E. Jove, J. Aveleira-Mata, H. Alaiz-Moretón, J.-L. Casteleiro-Roca, D. Y. M. del Blanco, F. Zayas-Gato, H. Quintián, and J. L. Calvo-Rolle, "Intelligent one-class classifiers for the development of an intrusion detection system: The MQTT case study," *Electronics*, vol. 11, no. 3, p. 422, Jan. 2022.

[16] J. A. Mata, A. L. M. Castañeda, M. T. G. Ordás, C. B. Cuellar, J. A. Benítez Andrades, and H. Alaiz Moreton, "IDS prototype for intrusion detection with machine learning models in IoT systems of the Industry 4.0," *DYNA*, vol. 96, no. 3, pp. 270–275, May 2021.

[17] S. Galli. (Mar. 2023). *Dealing With Imbalanced Datasets in Machine Learning: Techniques and Best Practices*. [Online]. Available: www.blog.trainindata.com/machine-learning-with-imbalanced-data/

[18] P. K. R. Maddikunta, M. Parimala, S. Koppu, T. R. Gadekallu, C. L. Chowdhary, and M. Alazab, "An effective feature engineering for DNN using hybrid PCA-GWO for intrusion detection in IoMT architecture," *Comput. Commun.*, vol. 160, pp. 139–149, Jul. 2020, doi: 10.1016/j.comcom.2020.05.048.

[19] S. Chesney and K. Roy, "AI empowered intrusion detection for MQTT networks," in *Proc. Int. Conf. Artif. Intell., Big Data, Comput. Data Commun. Syst. (icABCD)*, Aug. 2022, pp. 1–6.

[20] A. Haripriya and K. Kulothungan, "Secure-MQTT: An efficient fuzzy logic-based approach to detect DoS attack in MQTT protocol for Internet of Things," *EURASIP J. Wireless Commun. Netw.*, vol. 2019, no. 1, p. 90, Dec. 2019.

[21] . S. Kiran, R. K. Devisetty, N. P. Kalyan, K. Mukundini, and R. Karthi, "Building a intrusion detection system for IoT environment using machine learning techniques," *Proc. Comput. Sci.*, vol. 171, pp. 2372–2379, Jan. 2020.

[22] N. S. Alfaiz and S. M. Fati, "Enhanced credit card fraud detection model using machine learning," *Electronics*, vol. 11, no. 4, p. 662, Feb. 2022.

[23] A. Alhogail and I. Al-Turaiki, ''Improved detection of malicious domain names using gradient boosted machines and feature engineering,'' *Inf. Technol. Control*, vol. 51, no. 2, pp. 313–331, Jun. 2022.

[24] I. Al-Turaiki and N. Altwaijry, ''A convolutional neural network for improved anomaly-based network intrusion detection,'' *Big Data*, vol. 9, no. 3, pp. 233–252, Jun. 2021.

[25] M. Feurer and F. Hutter, ''Hyperparameter optimization,'' in *Automated Machine Learning: Methods, Systems, Challenges*, F. Hutter, L. Kotthoff, J. Vanschoren, Eds. Cham, Switzerland: Springer, 2019, pp. 3–33.

[26] M. J. Zaki and W. Meira, *Data Mining and Machine Learning: Fundamental Concepts and Algorithm*. Cambridge, U.K.: Cambridge Univ. Press, Jan. 2020.

[27] Y. Zhao, ''Chapter 5—Regression,'' in *R and Data Mining: Examples and Case Studies*. Amsterdam, The Netherlands: Academic Press, 2013, pp. 41–50. [Online]. Available: https://doi.org/10.1016/B978-0-12-396963-7.00005-2

[28] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, ''LIBLINEAR: A library for large linear classification,'' *J. Mach. Learn. Res.*, vol. 9, pp. 1871–1874, Jun. 2008.

[29] J. H. Friedman, ''Greedy function approximation: A gradient boosting machine,'' *Ann. Stat.*, vol. 29, no. 5, pp. 1189–1232, 2001.

[30] D. Berrar, ''Cross-Validation,'' *Oxford*, pp. 542–545, 2019. [Online]. ailable: https://doi.org/10.1016/B978-0-12-809633-8.20349-X

[31] N. W. S. Wardhani, M. Y. Rochayani, A. Iriany, A. D. Sulistyono, and P. Lestantyo, ''Cross-validation metrics for evaluating classification performance on imbalanced data,'' in *Proc. Int. Conf. Comput., Control, Informat. Appl. (ICINA)*, Oct. 2019, pp. 14–18.

[32] R. Mohammed, J. Rawashdeh, and M. Abdullah, ''Machine learning with oversampling and undersampling techniques: Overview study and experimental results,'' in *Proc. 11th Int. Conf. Inf. Commun. Syst. (ICICS)*, 2020, pp. 243–248.

[33] A. Y. C. Liu, ''The effect of oversampling and undersampling on classifying imbalanced text datasets,'' M.S. thesis, College Electr. Computer Eng., Univ. Texas Austin, Austin, TX, USA, 2004, doi: 10.26153/tsw/12300.

[34] R. Barandela, R. M. Valdovinos, J. S. Sánchez, and F. J. Ferri, ''The imbalanced training sample problem: Under or over sampling?'' in *Structural, Syntactic, and Statistical Pattern Recognition*. Berlin, Germany:, 2004, pp. 806–814.

[35] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, P. Prettenhofer, R. Weiss, and V. Dubourg, ''Scikit-learn: Machine learning in Python,'' *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Nov. 2011.