Received 16 January 2024, accepted 2 February 2024, date of publication 8 February 2024, date of current version 23 February 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3363889

RESEARCH ARTICLE

Realtime Feature Engineering for Anomaly Detection in IoT Based MQTT Networks

IMRAN[1,2], MEGAT F. ZUHAIRI[1], SYED MUBASHIR ALI[1,3], ZEESHAN SHAHID[4], MUHAMMAD MANSOOR ALAM[1,5,6,7], AND MAZLIHAM MOHD SU'UD[7]

[1]Malaysian Institute of Information Technology, Universiti Kuala Lumpur, Kuala Lumpur 50250, Malaysia
[2]Department of Computer Science, DHA Suffa University (DSU), Karachi, Sindh 75500, Pakistan
[3]Department of Software Engineering, Faculty of Computing, Muhammad Ali Jinnah University, Karachi 75400, Pakistan
[4]Electrical Engineering Department, Faculty of Engineering Practices and Sciences, Nazeer Hussain University, Karachi 75950, Pakistan
[5]Faculty of Computing, Riphah International University, Islamabad 46000, Pakistan
[6]Faculty of Engineering and Information Technology, School of Computer Science, University of Technology Sydney, Ultimo, NSW 2007, Australia
[7]Faculty of Computing and Informatics, Multimedia University, Cyberjaya 63100, Malaysia

Corresponding author: Mazliham Mohd Su'ud (mazliham@mmu.edu.my)

ABSTRACT The MQTTset dataset has been extensively investigated for enhancing anomaly detection in IoT-based systems, with a focus on identifying Denial of Service (DoS) attacks. The research addresses a critical gap in MQTT traffic anomaly detection by proposing the incorporation of the 'source' attribute from PCAP files and utilizing hand-crafted feature engineering techniques. Various filtering methods, including data conversion, attribute filtering, handling missing values, and scaling, are employed. Anomalies are categorized and prioritized based on frequency of occurrence, with a specific emphasis on DoS attacks. The study compares the performance of the decision tree and its eight variant models (ID3, C4.5, Random Forest, CatBoost, LightGBM, XGBoost, CART, and Gradient Boosting) for anomaly detection in IoT-based systems. Evaluation metrics such as prediction accuracy, F1 score, and computational times (training and testing) are utilized. Hyperparameter fine-tuning techniques like grid search and random search are applied to enhance model performance, accuracy, and reduce computational costs. Results indicate that the benchmark Decision Tree model achieved 92.57% accuracy and a 92.38% F1 score with training and testing times of 2.95 seconds and 0.86 seconds, respectively. The Feature Engineering (Modified) dataset demonstrated a substantial improvement, reaching 98.56% accuracy and a 98.50% F1 score, with comparable training and testing times of 0.70 seconds and 0.02 seconds. Furthermore, the Modified Decision Tree Algorithm significantly improved accuracy to 99.27%, F1 score to 99.26%, and reduced training time to 0.73 seconds and testing time to 0.14 seconds. The research contributes valuable insights into feature engineering and guides the selection of effective approaches for anomaly detection in IoT-based systems, providing early threat warnings and enhancing overall system security and reliability.

INDEX TERMS IoT, DoS, anomaly detection, MQTT.

I. INTRODUCTION

"Knowledge is power" [1]. The power of knowledge is an important element for both individuals and groups today and in the future for maintaining a valued legacy, learning new things, solving issues, developing core competencies, and beginning new circumstances such as AI [2], the IoT [3], and many other things have become feasible [4].

The associate editor coordinating the review of this manuscript and approving it for publication was Chien-Ming Chen.

Modern technology is seeing a massive increase in IT toward the IoT [4]. The research findings indicated prospective growth prediction from 2019 to 2030 are in line with the technical objectives that have the potential to revolutionize the interconnected world, as shown in Figure 1 [5].

Table 1 include in the research article to provide a list of acronyms used throughout this paper. The IoT Security-Focused Datasets are specifically curated for studying and analyzing security aspects related to IoT devices and systems [6]. IoT refers to the interconnected network

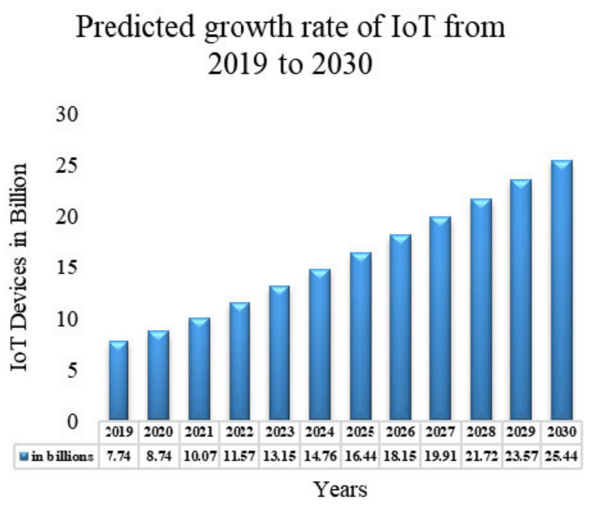


FIGURE 1. The average growth prediction in IoT from 2019 to 2030.

of physical devices, vehicles, buildings, and other items embedded with electronics, software, sensors, and network connectivity [7]. Parameters Analyzed refer to various aspects or attributes of the data that are being examined. The parameters include network traffic patterns, device behavior, vulnerabilities, attack patterns, and others [8].

Dataset names refer to the names given to these datasets, which are used to identify and reference them. Each dataset might focus on specific aspects of IoT security and may have a unique name. Respective years of creation indicate the years in which the datasets were created or compiled. This information provides context for understanding the relevancy and currency of the data within the datasets. In simulated environments, IoT security datasets often include simulated environments that mimic real-world IoT networks and scenarios. [9].

Such environments allow researchers and analysts to study security threats and vulnerabilities in controlled settings. Associated PCAP Files contain captured network traffic data, including the actual packets of information exchanged between devices on a network [10]. The files are crucial for analyzing network behavior, identifying patterns, and detecting potential security issues [11]. Dataset Sizes refers to the amount of data present in each dataset, typically measured in terms of storage size (e.g., megabytes, gigabytes, terabytes). Dataset size can impact the depth and scope of analyses that can be performed [12]. Each dataset may have unique characteristics that make it suitable for certain types of analysis. The characteristics include the types of IoT devices covered, the variety of attacks simulated, and the complexity of the network topology [13]. In other words, in the dynamic landscape of IoT-based systems, the dataset has emerged as a pivotal focal point, captivating the attention of researchers delving into the intricate realm of anomaly detection [14]. This study ventures into the refinement of anomaly detection techniques, orchestrating a symphony of various filtering

TABLE 1. List of acronyms.

S. No.	acronyms	Definition
1	MQTT	Message Queuing Telemetry Transport
2	DoS	Denial-of-Service
3	DDoS	Distributed Denial of Service
4	ID3	Iterative Dichotomiser 3
5	CART	Classification and Regression Tree
6	IoT	Internet of Things
7	AI	Artificial Intelligence
8	IT	Information Technology
9	PCAP	Packet Capture
10	CICIDS	Canadian Institute for Cybersecurity Intrusion Detection Dataset
11	N-BaIoT	Network-Based Internet of Things
12	Bot	Robot or Malicious Software Agent
13	IDS	Intrusion Detection System
14	SlowITe	Slow Internet of Things Exploits
15	DT	Decision Tree
16	ANN	Artificial Neural Network
17	NB	Naïve Bayes
18	RF	Random Forest
19	MLP	Multi-Layer Perception
20	LR	Logistic Regression
21	GB	Gradient Boost
22	ML	Machine Learning
23	KNN	K-Nearest Neighbors
24	DNN	Deep Neural Network
25	IP	Internet Protocol
26	Colab	Google Colaboratory
27	GOSS	Gradient-Based One-Side Sampling
1	MQTT	Message Queuing Telemetry Transport
2	DoS	Denial-of-Service
3	DDoS	Distributed Denial of Service
4	ID3	Iterative Dichotomiser 3

methods ranging from data conversion and attribute filtering to handling missing values and scaling. With a primary objective of elevating the identification precision of anomalies, the research casts a spotlight on the detection of attacks within the IoT environment. Such an overview would be useful for researchers, analysts, and practitioners interested in studying IoT security threats and vulnerabilities in a controlled environment.

The ‘source’ (IP addresses) plays a significant role in understanding the patterns and behaviors of devices on a network. However, the existing work on predicting the anomalies in MQTTset dataset, the ‘source’ (IP address) has been ignored.

Following are the key contributions.

- By identifying a gap in the dataset regarding the ‘source’ attribute, the research employs hand-crafted feature engineering techniques to incorporate this attribute from PCAP files. This addition enhances the data analysis and improves the prediction accuracy and prediction time in the MQTTset dataset, particularly for identifying Denial of Service (DoS) attacks.

- The research conducts benchmarking analysis, comparing the performance of different decision tree and variant models, including ID3, C4.5, Random Forest, CatBoost, LightGBM, XGBoost, and Gradient Boosting, on the same MQTTset dataset. This comprehensive evaluation allows for the identification of the most effective approach for detecting anomalies in IoT-based systems.
- The research further refines the selected model's performance by fine-tuning its hyperparameters. By systematically adjusting the values of hyperparameters and evaluating the model's performance, the research achieves improved accuracy, precision, recall, and F1-score, ultimately enhancing the reliability of the anomaly detection system. Incorporated automated Hyperparameter tuning to eliminate human dependency on the prediction models.

In summary, this research significantly advances the state-of-the-art approaches in IoT-based anomaly detection and achieving substantial performance improvements. In addition, the research emphasizes the importance of computational efficiency, the new approach resulted in effective anomaly prediction without imposing excessive computational overhead. The subsequent sections of this paper are structured as follows: In Section II, present a survey of recent works closely related to the topic. Section III outlines the methodology, architectural setup and introduces the ML techniques employed in the proposed approach. The experimental testbed, along with the results and subsequent discussions, is elaborated upon in Section IV. Finally, Section V provides a conclusion to this paper.

II. RELATED WORK

The IoT is made up of a large variety of intelligent gadgets that can gather, store, analyze, and communicate data. The IoT environment is difficult to protect due to resource limits, heterogeneity, and the dispersed nature of smart devices. This rendered network protection techniques such as anti-malware and anti-virus ineffective. Such issues, as well as the nature of IoT applications, require the implementation of a monitoring system, such as anomaly detection, at both the device and network levels, outside the organizational border. It implies that an anomaly detection system is better positioned than any other security technique to safeguard IoT devices [15].

A. ANOMALY DETECTION IN IoT

Today's modern world is populated by a massive amount of IoT devices that generate massive amounts of data, and anomaly detection are a part of practically every system. The anomalies could indicate resource waste in an industrial system, a critical condition in a spacecraft system to avoid unanticipated issues, or identifying anomalous behavior in medical devices. As a result, being able to detect abnormalities can have a significant impact on the entire efficiency of any examined system [16]. Defining the precise limits between normal and abnormal actions is one of the most difficult tasks in detecting abnormalities. When compared to normal behaviors, deviant behavioral patterns are extremely rare in real-life circumstances. Most models in anomaly detection approaches are trained on different patterns of normal activity. Any occurrence that deviates from this is deemed anomalous behavior [17].

TABLE 2. Characteristics of IoT attacks [19].

Categorie s	# of Training examples	# of Testing examples	Total	# of attr.	# of cls
Legitimate	115,824	49,651	165,475		
DoS	91,156	39,077	1,30,233		
Bruteforce	10,150	4,351	14,501	34	6
Malformed	7,646	3,278	10,924		
SlowITe	6,441	2,761	9,202		
Flood	429	184	613		

B. THE MQTT PROTOCOL

MQTT is an international communications standard for the IoT. It is intended to be a very lightweight publish/subscribe message carrier for linking wireless connections with a tiny code footprint and low network traffic. MQTT is now utilized in a broad range of sectors, including automotive, manufacturing, telecommunications, oil and gas [18]. The capability for continuous sessions in MQTT minimizes the time it takes to reconnect the customer to the broker.

C. MQTTSET DATASET

MQTTset dataset [19] is a comprehensive collection of network traffic data that captures various types of cyber-attacks. Among the different attack types included in the dataset, five anomalies stand out, each with its distinct characteristics. It combines legitimate and anomalous data, including DoS, brute-force, flood, malformed, and SlowITe attacks against the MQTT network. MQTTset can be utilized to train ML models for the implementation of detection systems capable of protecting IoT environments. The following describes the type of attacks in the '*MQTTset*' dataset. Table 2' shows the comprehensive examination of the '*MQTTset*' dataset, a dataset consisting of 34 attributes in total. Within this dataset, particular attention is given to the 'target' attribute, which exhibits a diverse distribution comprising six distinct classes. This study aims to provide a thorough analysis of the '*MQTTset*' dataset, with a primary focus on understanding the class distribution patterns within the 'target' attribute. This paragraph establishes the context for our research investigation, outlining the dataset's key features and its pivotal 'target' attribute's multi-class composition.

DoS attacks aim to disrupt the availability of a network or service by overwhelming it with a high volume of requests or malicious activity. Within the '*target*' attribute of our dataset, we have identified a total of six distinct classes. In particular, the '*DoS*' class has garnered our attention, consisting of 130,233 rows in total. This '*DoS*' class has been further partitioned into 91,156 instances designated for training and

39,077 instances reserved for testing purposes see Table 2. This paragraph serves to provide a structured overview of the class distribution within the 'target' attribute, with a specific focus on the 'DOS' class and its respective training and testing subsets.

Bruteforce attacks involve systematically trying multiple combinations of usernames and passwords to gain unauthorized access to a system or account. See in Table 2 the 'target' attribute of our dataset, we have identified a total of six distinct classes, each representing different data categories. In particular, the *'bruteforce'* class has been a subject of interest, comprising 14,501 instances. This *'bruteforce'* class has been further segregated into 10,150 instances allocated for training and 4,351 instances earmarked for testing purposes.

Malformed packets or messages refer to data that does not adhere to the expected format or protocol specification. Instances in this class may indicate anomalies caused by incorrectly formatted or structured data. Within the '*target*' attribute of our dataset, we have identified a total of six distinct classes, each representing unique data categories. Notably, Table 2 shows the 'malformed' class has captured our attention, comprising a total of 10,924 instances. Further examination reveals that the 'malformed' class has been partitioned into 7,646 instances designated for training, while 3,278 instances have been reserved for testing purposes.

SlowITe attacks involve intentionally slowing down network communication or specific services to disrupt their normal functioning. Table 2 shows the 'target' attribute of our dataset, we have identified a set of six distinct classes, each signifying specific data categories. Notably, the '*slowite*' class has piqued our interest, comprising a total of 9,202 rows. Delving deeper into this class, it is deemed that the dataset has been categorized into 6,441 instances allocated for training and 2,761 instances earmarked for testing purposes.

Flood attacks aim to overwhelm a network or system by flooding it with a large volume of traffic, often exceeding its capacity. Table 2 shows the 'target' attribute of our dataset, we have identified a total of six distinct classes, each representing specific data categories.

Notably, the '*flood*' class has garnered our attention, encompassing a total of 613 instances. A more detailed examination reveals that this 'flood' class has been further categorized into 429 instances designated for training and 184 instances allocated for testing purposes.

Legitimate class represents normal and legitimate network traffic or behavior. Instances belonging to this class are expected to exhibit normal patterns and are considered non-anomalous. Table 2 shows the number of training and testing examples used in each category. The 'target' attribute in the dataset under scrutiny exhibits a diverse set of classes. This research work will focus on one of the classes, 'legitimate,' which consists of 165,475 instances. The class 'legitimate' is further divided into 115,824 training examples and 49,651 testing examples.

Table 4 presents a the IoT security-focused dataset. It is a comprehensive overview of the parameters analyzed within recently accessible IoT security-focused datasets. These datasets encompass essential information such as dataset names, respective years of creation, simulated environments, associated PCAP files, dataset sizes, and specific dataset characteristics. The table seems to cover various parameters related to the datasets, such as names, creation years, simulated environments, PCAP files, sizes, and specific characteristics.

TABLE 3. Association between MQTT attacks and features [30].

Attacks	Source Port	TCP Length	Message Type	Keep Alive	Conn. ACK
DoS	✓	---	---	---	---
Flood	---	✓	✓	---	---
SlowITe	---	---	---	✓	---
Malformed	---	---	✓	---	---
Brute-force	✓	---	---	---	✓
DoS	✓	---	---	---	---

D. SIGNIFICANT FEATURES

The attacker needs to identify the target system they want to disrupt. The IP address is used to uniquely identify a device on a network, enabling the attacker to pinpoint their target. According to [19], some features such as source/destination and port address can play important role in detecting attacks, however these features were ignored in their experiments. In another study by [30], the authors analyzed the significance of the features as evident in Table 3.

E. ML IN MQTT NETWORKS

This section discusses the recent references related to the DT algorithm. ML algorithms have been widely used in MQTT Networks.

1) BENCHMARK ALGORITHM

Supervised ML algorithms such as DT, RF, ANN, NB, and MLP are widely used for the prediction of attacks. Figure 2 shows the overall process of training and evaluation on MQTTset [19].

The literature reveals that DT is the best performing algorithm (winning algorithm) DT when applied to MQTTset dataset. The algorithm creates a tree-like model of decisions and their possible consequences, with each internal node representing a decision based on a feature and each leaf node representing a class label or a numerical value [31]. Each node in the DT may balance potential actions against each other based on their benefits, costs, and probability.

In the context of the DT model, the performance metrics are outlined in Table 5. The achieved accuracy rate is notably high at 92.57%, highlighting the model's precision in making accurate predictions. The F1 score, a balanced measure of precision and recall is 92.38%. Considering computational aspects, the training process took 2.95 seconds, while the subsequent testing phase consumed only 0.86 seconds. This

TABLE 4. Recent available IoT security-focused datasets.

Sno	Dataset	Ref	Year	Simulated Environment	Data (PCAP file)	Size	Details
1	CICIDS2017	[20]	2017	✓	✓	884.65MB	This dataset contains a variety of network traffic scenarios, including both normal and malicious activities, to mimic real-world network environments.
2	N-BaIoT	[21]	2018	✓	✓	8GB	Network traffic datasets are often used to study network behavior, develop IDS, and analyze the activities of botnets and other malicious actors.
3	Kitsune Network Attack	[22]	2018	✓	✓	64.18GB	This Dataset likely aims to provide a collection of network traffic data to facilitate research, and evaluation of IDS, anomaly detection methods, and other security-related technologies for IoT networks.
4	IoT Security	[23]	2018	✓	---	132MB	This dataset involving the IoT devices would likely aim to provide researchers with a collection of network traffic data to study the security of IoT devices in a controlled environment.
5	IoT Network Intrusion	[24]	2019	✓	✓	823.69MB	This Dataset involving simulated networks with attacks like scans and DoS aims to provide researchers and security professionals with a collection of network traffic data to study and improve the detection and mitigation of IoT-related cyber threats.
6	Bot-IoT	[25]	2019	✓	---	259.72MB	This Dataset likely aims to provide a collection of network traffic data representing a realistic IoT environment where attacks such as data and keylogging are executed. This dataset would be valuable for researching and developing techniques to detect and mitigate advanced cyber threats targeting IoT devices.
7	IoT-23	[26]	2020	✓	✓	21GB	The "IoT-23" dataset is a collection of network traffic data in a smart home scenario. This dataset is designed to encompass both infected (compromised or malicious) and non-infected (benign) IoT devices within a smart home network environment.
8	IoT DoS and DDoS Attack	[27]	2020	✓	✓	487MB	Creating an IoT, DoS and DDoS attack dataset that transforms network traffic into images and uses a CNN model for improved analysis is an interesting approach for studying and mitigating these types of cyber threats. By capturing and transforming network traffic data or explore existing datasets that can be adapted to this approach.
9	MQTT-IoT-IDS2020	[28]	2020	✓	✓	1.65GB	This dataset likely pertains to a collection of network traffic data in a simulated MQTT network architecture. The dataset is designed to include various attack scenarios executed within this simulated environment to facilitate the development and evaluation of IDS for MQTT-based IoT networks.
10	**MQTTset**	[29]	**2020**	✓	✓	**10GB**	**This dataset designed to simulate a MQTT network architecture with various attacks executed within the simulated environment. This type of dataset could serve as a valuable resource for researchers and security professionals to develop, test, and evaluate IDS and security mechanisms for MQTT-based IoT networks.**

TABLE 5. The results of MQTTset (Reduced) dataset using benchmark algorithms.

Algos	Accuracy (%)	F1 score (%)	Training Time (SEC)	Testing Time (SEC)
NB	79.47%	76.51%	1.39	1.08
MLP	82.77%	81.73%	338.56	0.57
ANN	81.83%	80.62%	159.49	10.42
LR	79.71%	77.02%	30.70	0.14
DT	**92.57%**	**92.38%**	**2.95**	**0.86**
RF	92.56%	92.38%	15.13	1.42
DNN	82.72%	81.60%	202.58	6.08
KNN	91.32%	91.23%	0.41	548.42

signifies the efficiency of the DT model in swiftly processing data during both training and testing stages.

The RF [32] algorithm is a supervised classification method builds upon group of DT. The RF has improved execution speed as compare with DT. Within the context of the RF model, the achieved performance metrics such as the model demonstrates a commendable accuracy level of 92.56%, indicating its proficiency in making precise predictions, The F1 score, which harmonizes precision and recall is 92.38%, underscoring the model's balanced predictive capabilities. Regarding computational efficiency the training process spans 15.13 seconds, reflecting the time taken to develop the RF model based on the provided data and during the testing phase, the model showcases a testing time of 1.42 seconds, showcasing its efficiency in promptly evaluating data.

NB [33] is a popular ML algorithm based on Bayes' theorem, which is also used for classification tasks. It assumes that the features are independent of each other, hence the name "naïve" [34]. The NB is a classic classifier that uses Bayes' theorem of pre-probability to categorize data instances. It offers a quick training pace for both small and large datasets. It is less susceptible to missing data, but it requires previous probabilities to be calculated. Table 5 presents the performance metrics for the conducted experiment. The accuracy achieved is 79.47%, showcasing the model's ability to make correct predictions. Additionally, the F1 score, indicating a balanced measure between precision and recall is, 76.51%. In terms of computational efficiency, the training process time 1.39 seconds, while the testing phase required 1.08 seconds. This demonstrates the rapid processing speed of the model during both training and testing, underlining its efficiency in handling data.

LR [35] is a statistical method used for binary classification tasks in ML. This technique is typically used for classification rather than regression. LR models the probability of a binary outcome (such as 0 or 1) based on one or more predictor variables. Table 5, the LR model, the experiment's outcomes are presented such as the model achieves an accuracy of 79.71%, showcasing its capability to make accurate predictions. The F1 score, at 77.02%, signifies a balanced amalgamation of precision and recall, illustrating the model's comprehensive predictive performance. The training process takes 30.70 seconds, indicating the duration required to establish the model's predictive framework. During the testing phase, the model

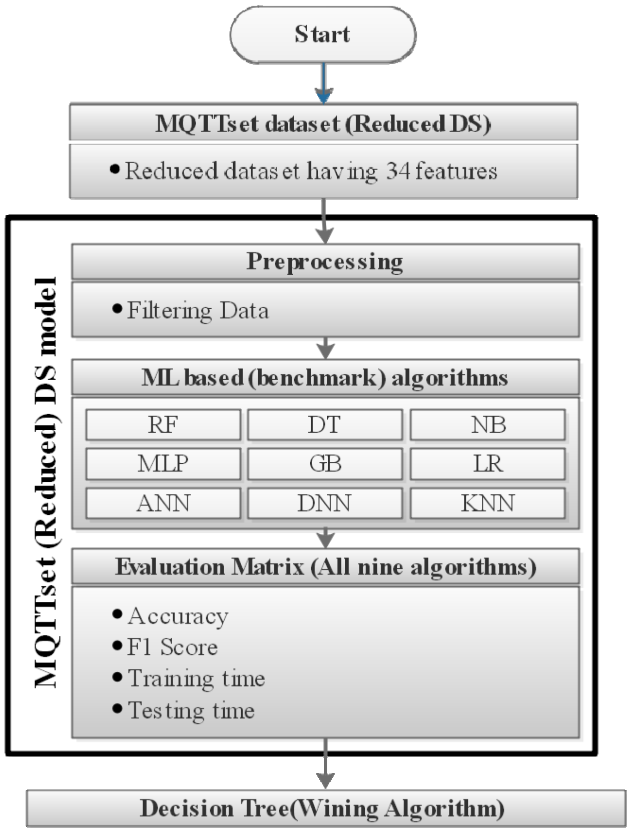

Start
MQTTset dataset (Reduced DS)
Reduced dataset having 34 features
MQTTset (Reduced) DS model
Preprocessing
Filtering Data
ML based (benchmark) algorithms
RF
DT
NB
MLP
GB
LR
ANN
DNN
KNN
Evaluation Matrix (All nine algorithms)
Accuracy
F1 Score
Training time
Testing time
Decision Tree(Wining Algorithm)

FIGURE 2. MQTTset (reduced) dataset model using benchmark algorithms.

efficiently evaluates data within a mere 0.14 seconds, underlining its swift and effective assessment abilities.

MLP [36] is the model employs a multi-layer structure, consisting of interconnected neurons organized into input, hidden, and output layers. Each neuron in a layer is connected to every neuron in the subsequent layer, forming a network that can capture intricate relationships within the data. Table 5 shows the result of MLP model, the experiment's outcomes are presented systematically model performance such as the model achieves an accuracy rate of 82.77%, showcasing its ability to make correct predictions across the dataset. With an F1 score of 81.73%, the model demonstrates a balanced blend of precision and recall, emphasizing its capability to effectively handle positive instances and avoid false positives. On the contrary, the training process takes a total of 338.56 seconds, indicating the duration required for the model to learn from the training data and optimize its parameters and the testing phase, the model efficiently evaluates new data in just 0.57 seconds, highlighting its rapid and effective classification abilities.

ANN [37] exhibit parallel processing, non-linearity, learning capabilities, and the ability to handle various data types. It can automatically extract features, generalize from data, and tackle complex tasks. However, the complexity and interpretability challenges should be considered when applying them to different applications. The characteristics of the ANN model are highlighted in structured manner based on the provided experiment results see Table 5. In the model performance the ANN achieves an accuracy of 81.83%, indicating its proficiency in making accurate predictions on the dataset. The F1 score is 80.62%, showcases the model's ability to achieve a balanced trade-off between precision and recall in predictions, In the computational efficiency, the training process requires a total of 159.49 seconds, reflecting the duration needed for the model to learn from the training data and optimize its internal parameters and the testing phase, the model evaluates new data in 10.42 seconds, highlighting its ability to process and classify data within a reasonable timeframe.

KNN [39] is an intuitive ML algorithm used for classification and regression tasks. It falls under the category of supervised learning algorithms and is a part of the instance-based learning family. The main idea behind KNN is to classify or predict a new data point based on the majority class or average value of its KNN in the feature space. See Table 5 shows the performance metrics, the KNN algorithm achieves an accuracy of 91.32%, indicating its proficiency in correctly classifying data points. With an F1 score of 91.23%, the KNN algorithm demonstrates a balanced combination of precision and recall in its predictions. The KNN algorithm doesn't have a traditional training phase. Instead, it stores the training data, so the training time is minimal at 0.41 seconds. During the testing phase, the KNN algorithm takes 548.42 seconds to classify new data points based on their proximity to the training instances.

DNNs [38] consist of multiple hidden layers that enable them to capture intricate data relationships. They are designed to handle complex patterns and representations. The provided experiment results see Table 5 reveal the DNN's high accuracy and balanced F1 score, as well as its training and testing efficiency. According to the performance metrics, The DNN achieves an accuracy of 82.72%, showcasing its proficiency in making accurate predictions across the dataset. With an F1 score of 81.60%, the DNN demonstrates its capability to achieve a balanced blend of precision and recall in predictions. The training process of the DNN takes a total of 202.58 seconds. This time frame encompasses the model's learning phase, during which it adapts its internal parameters to the provided training data. During the testing phase, the DNN efficiently evaluates new data in 6.08 seconds, highlighting its ability to process and classify data with speed.

Table 5 shows the predictive performances, F1-Score, training and testing time on MQTTset reduced dataset [19] by applying the above ML algorithms. The F1-score is a metric commonly used to evaluate the performance of classification models, particularly in situations where there's an imbalance between the classes.

As it is evident from the Table 5 results, the DT achieved the highest predictive accuracy and computational time in term of testing and training time, therefore the DT algorithms

TABLE 6. The results of MQTTset (Reduced) dataset using variants of DT algorithms.

Variants of DT Algorithms	Accuracy (%)	F1 score (%)	Training Time (SEC)	Testing Time (SEC)
DT	**92.57%**	**92.38%**	**2.95**	**0.86**
RF	92.56%	92.38%	15.13	1.42
ID3	92.55%	92.37%	3.11	0.83
CatBoost	92.80%	92.62%	43.12	0.87
LightGBM	92.80%	92.63%	20.93	4.50
XGBoost	92.54%	92.36%	14.65	1.37
C4.5	92.56%	92.37%	4.37	0.40
GB	92.55%	92.33%	206.38	1.54
CART	92.56%	92.38%	1.38	0.23

has been chosen for further modification to achieve improvement on MQTTset dataset.

2) VARIANTS OF DT ALGORITHM

The following section describes the variants of DT algorithm.

ID3 [40]: ID3 is one of the earliest DT algorithms. It uses the concept of information gain to select the best feature at each step of the tree construction process. ID3 is primarily used for classification tasks and works well with categorical data. Within the context of the ID3 classifier, Table 6 shows the achieved performance metrics, where the model demonstrates an accuracy level of 92.55%, indicating its proficiency in making precise predictions, The F1 score, which harmonizes precision and recall, stands at an impressive 92.37%, underscoring the model's balanced predictive capabilities. Regarding computational efficiency the training process spans 0.83 seconds, reflecting the time taken to develop the ID3 classifier based on the provided data and during the testing phase, the model showcases a testing time of 1.42 seconds, showcasing its efficiency in promptly evaluating data.

C4.5 [41]: is an extension of the ID3 algorithm. It introduces the concept of gain ratio, which addresses the bias of ID3 towards attributes with a large number of distinct values. C4.5 can handle both categorical and numerical features and supports missing values. Table 6 presents the performance metrics for the conducted experiment. The accuracy achieved is 92.56%, showcasing the model's ability to make correct predictions. Additionally, the F1 score, indicating a balanced measure between precision and recall, stands at 92.37%. In terms of computational efficiency, the training process took 4.37 seconds, while the testing phase required 0.40 seconds. This demonstrates the rapid processing speed of the model during both training and testing, underlining its efficiency in handling data.

LightGBM Classifier [42]: is another GB framework that focuses on high efficiency and scalability. It uses a technique called GOSS to select and prioritize instances during tree construction, reducing memory usage and improving training speed. LightGBM also supports categorical features and provides excellent performance on large datasets. Table 6 shows about the result of the LightGBM ML algorithm to tackle a classification task. The results were quite promising, with the model achieving an accuracy of 92.80%. This indicates the model's proficiency in correctly categorizing instances from the dataset. The F1 score, a metric that balances precision and recall, was also impressive at 92.63%. Efficiency was another notable aspect of our study. The training process, which involved teaching the model on the dataset, took a mere 20.93 seconds. This signifies the algorithm's capability to swiftly learn patterns and relationships within the data. Subsequently, during the testing phase, where the model's generalization to unseen data was evaluated, the process took only 4.50 seconds.

XGBoost Classifier [43]: is an optimized GB framework that has gained popularity for its performance and flexibility. It uses a combination of GB and regularization techniques to build DT. XGBoost includes various advanced features like parallel processing, handling missing values, and handling sparse data, making it widely used in ML competitions. Table 6 shows about the result of the XGBoost classifier for a classification endeavor. The outcomes were notably encouraging, as the model attained an accuracy of 92.54%. This underscores the model's competence in effectively classifying instances within the dataset. The F1 score for XGBoost is 92.36%. Efficiency emerged as another salient aspect of our investigation. The training phase, encompassing the model's learning process on the dataset, was completed within a mere 14.65 seconds. This signifies the classifier's rapid assimilation of patterns and information present in the data. Subsequent to training, the testing phase, which gauged the model's performance on previously unseen data, lasted just 1.37 seconds.

CatBoost Classifier [44]: is a GB framework that specifically addresses categorical feature handling. It incorporates a novel approach to deal with categorical variables, providing better accuracy and training efficiency. CatBoost utilizes a combination of ordered boosting and categorical boosting to build DT. Table 6 shows the result of CatBoost classifier, the experiment's outcomes are presented systematically model performance such as the model achieves an accuracy rate of 92.80%, showcasing its ability to make correct predictions across the dataset. With an F1 score of 92.62%, the model demonstrates a balanced blend of precision and recall, emphasizing its capability to effectively handle positive instances and avoid false positives. In other side, the training process takes a total of 43.12 seconds, indicating the duration required for the model to learn from the training data and optimize its parameters and the testing phase, the model efficiently evaluates new data in just 0.87 seconds, highlighting its rapid and effective classification abilities.

GB classifier [45]: is a popular ML technique used for both regression and classification tasks. It's an ensemble learning method that combines the predictions of multiple weak learners (usually DTs) to create a strong predictive model. GB builds the model in a stage-wise manner, where each new model corrects the errors of the previous ones. Table 6 shows about the result of the power of the GB Classifier for a critical classification task. The results obtained were notably compelling, with the model achieving an accuracy of 92.55%. This accomplishment underscores the classifier's adeptness in effectively categorizing instances within the dataset. The F1 score is 92.33%. Of particular interest was the efficiency aspect of this investigation. The training phase, encompassing the rigorous learning process of the model on the dataset, extended over a duration of 206.38 seconds. This underscores the comprehensive nature of the model's learning process, which may contribute to its nuanced performance. Following the training phase, the testing stage, aimed at evaluating the model's generalization to new data, concluded within a succinct 1.54 seconds.

The above variants of DT have been applied in the experiments results available in section IV

III. PROPOSED APPROACH

The research presented makes several significant contributions to the field of anomaly detection in IoT-based systems using the MQTTset dataset. The research applies various filtering techniques, such as data conversion, attribute filtering, handling missing values, label encoding, and feature selection to improve the quality and structure of the dataset. This ensures that the subsequent analysis and modeling steps are conducted on reliable and meaningful data. By identifying a gap in the dataset regarding the 'source' attribute, the research employs hand-crafted feature engineering techniques to incorporate this attribute from PCAP files. This addition enhances the data analysis and improves the accuracy of anomaly detection, particularly for identifying DoS attacks.

The research conducts benchmarking analysis, comparing the performance of different DT and variant models, including ID3, C4.5, RF, CatBoost, LightGBM, XGBoost, and GB, on MQTTset dataset. This comprehensive evaluation allows identification of the most effective approach for detecting anomalies in IoT-based systems.

The research further refines the selected performance of the model by fine-tuning the hyperparameters. By systematically adjusting the values of hyperparameters and evaluating the model's performance, the research achieves improved accuracy, precision, recall, and F1-score, ultimately enhancing the reliability of the anomaly detection system.

A. METHODOLOGY

In this research work on the exploration of anomalies within IoT-based systems using the MQTTset dataset, a comprehensive approach was taken. This involved multiple stages, starting with data preparation as the initial step to refine the raw information for analysis. Subsequent phases encompassed the meticulous cleaning and transformation of the data into a structured format suitable for machine learning algorithms. A key focus of this research was the determination of the significance of 'source' (IP addresses) through the application of handcrafted feature engineering methods. This

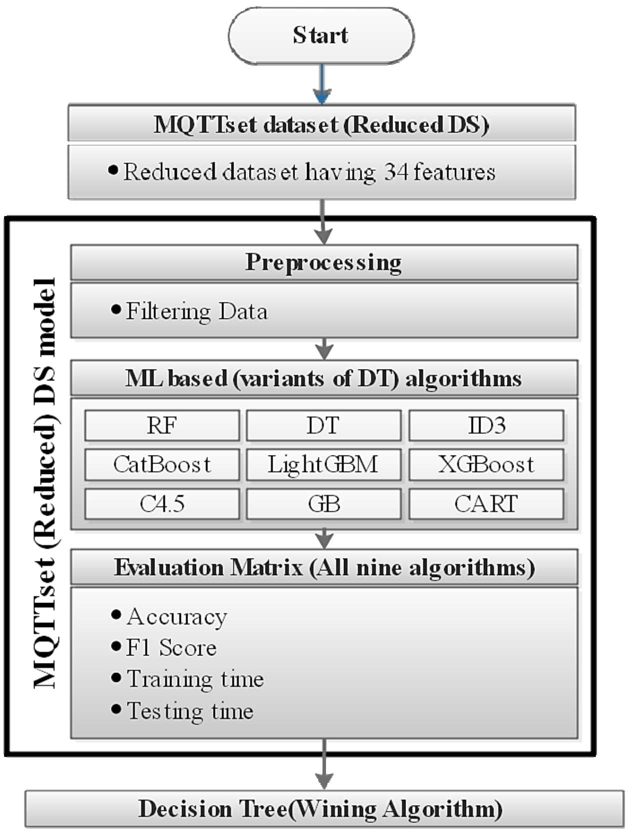

Start
MQTTset dataset (Reduced DS)
• Reduced dataset having 34 features
MQTTset (Reduced) DS model
Preprocessing
• Filtering Data
ML based (variants of DT) algorithms
RF
DT
ID3
CatBoost
LightGBM
XGBoost
C4.5
GB
CART
Evaluation Matrix (All nine algorithms)
• Accuracy
• F1 Score
• Training time
• Testing time
Decision Tree(Wining Algorithm)

FIGURE 3. MQTTset (reduced) dataset model using variants of DT algorithms.

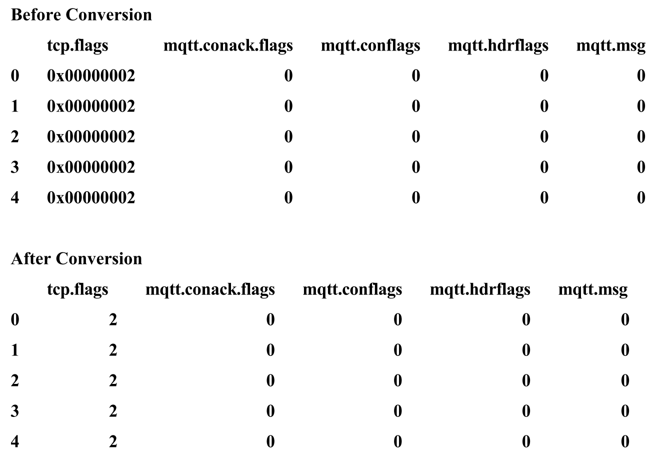
not only aimed to enhance prediction accuracy but also sought to optimize prediction time within the MQTTset dataset. Additionally, the integration of automated hyperparameter tuning played a crucial role in eliminating human dependency on the prediction models, further advancing the efficiency and robustness of the analytical process. The overall methodology has been presented in Figure 8.

B. PREPROCESSING TECHNIQUES

The goal of preprocessing is to prepare the data in a way that enhances the performance and effectiveness of the model. The following preprocessing techniques has been applied.

1) HEXADECIMAL CONVERSION

Hexadecimal data can contain valuable information. For instance, in network traffic analysis, hexadecimal values might represent packet headers or flags. Converting the hex values to binary or integers can help extract meaningful features for ML models to detect anomalies or patterns. Figure 4 shows the hexadecimal to numeric conversion is applied to the attributes (***'tcp.flags'***, ***'mqtt.conack.flags'***, ***'mqtt.conflags'***, ***'mqtt.hdrflags'***, ***'mqtt.msg'***). The ***'tcp.flags'*** attribute likely represents the flags in a TCP (Transmission Control Protocol) packet header. Each flag can be represented by a specific bit in binary, ***'mqtt.conack.flags'***,



Before Conversion

	tcp.flags	mqtt.conack.flags	mqtt.conflags	mqtt.hdrflags	mqtt.msg
0	0x00000002	0	0	0	0
1	0x00000002	0	0	0	0
2	0x00000002	0	0	0	0
3	0x00000002	0	0	0	0
4	0x00000002	0	0	0	0

After Conversion

	tcp.flags	mqtt.conack.flags	mqtt.conflags	mqtt.hdrflags	mqtt.msg
0	2	0	0	0	0
1	2	0	0	0	0
2	2	0	0	0	0
3	2	0	0	0	0
4	2	0	0	0	0

FIGURE 4. Before and after conversion of 'flag' attributes.

'mqtt.conflags', 'mqtt.hdrflags', 'mqtt.msg' attributes are related to the MQTT (Message Queuing Telemetry Transport) protocol, commonly used in IoT. Converting hex values to numeric ones allows for easier interpretation of MQTT packet details. ML models often require numerical input. By converting hex values to binary or integers, the data becomes suitable for input into these models. Anomalies or patterns in network traffic can be detected by training the ML models on these converted numeric features.

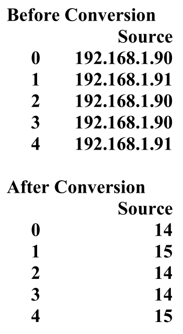
2) LABEL ENCODING

This technique is used to transform categorical data, specifically in the context of IP addresses that is suitable for ML algorithms. In binary encoding, each category is represented by a binary number, and each bit in the binary number corresponds to a specific category or feature. For IP addresses, this means breaking down each part of the address (e.g., octets in IPv4) into its binary representation. For example, an IP address like 192.168.1.90 might be represented as a binary vector where each octet is converted into an 8-bit binary number (byte).

As for label encoding, it's a method used to convert categorical data into numerical format by assigning a unique numerical label to each category. In Figure 5, it could involve assigning numerical labels to different parts of the IP address, such as octets or specific ranges of values.

3) HANDLING MISSING VALUES

Dealing with missing values is indeed crucial in data preprocessing for ML. The methods—dropping rows with null values and replacing specific values—are common strategies. Dropping rows with null values is a technique that involves removing entire rows from the dataset if they contain any null values. This operation eliminates rows with any missing values. Instead of dropping rows, it might choose to replace missing values with a specific value. This is often done to retain as much data as possible. The replacing 'MQTT' with '1' in the *'mqtt.protoname'* attribute of a DataFrame named 'df'. These techniques help ensure that the dataset remains usable for training machine learning models. However, it's essential to carefully consider the implications of these choices, as they can impact the model's performance. Provide more details on the above filtering techniques used and the specific methods employed for data conversion.



Before Conversion

	Source
0	192.168.1.90
1	192.168.1.91
2	192.168.1.90
3	192.168.1.90
4	192.168.1.91

After Conversion

	Source
0	14
1	15
2	14
3	14
4	15

FIGURE 5. Before and after label encoding technique.

4) FEATURE SELECTION

It seems like in the context of the MQTTset dataset, the focus was on feature selection to streamline the dataset and remove unnecessary complexity. The features *'mqtt.msg', 'Date', and 'Time'* were specifically mentioned as being removed. Feature selection techniques aim to choose the most relevant features for a ML model while discarding irrelevant or redundant ones. The feature selection techniques in the MQTTset dataset focused more on data preprocessing and filtering based on specific criteria.

C. FEATURE ENGINEERING

This research endeavors to enhance the depth of data analysis within the 'MQTTset' dataset by incorporating a crucial attribute, namely 'source,' sourced from the PCAP file. The benchmark MQTTset (reduced) dataset is missing the key attribute 'source', however such attribute is available in the raw PCAP file provided by the same authors. To extract this feature, a handcrafted feature engineering technique has been applied see Figure 6. Handcrafted features, often referred to as hand-engineered features, are specific attributes or variables created by domain experts or data scientists through manual design and feature extraction techniques.

The features are derived from the original data by applying expert knowledge, domain-specific insights, or mathematical transformations.

It involves manually designing and extracting the most relevant features from the PCAP file. To address the existing gap in the dataset, we have employed a meticulous approach to feature engineering.

By introducing the 'source' attribute, our aim is to augment the dataset's analytical capacity significantly see Figure 7.

The literature review underscores the pivotal importance of including this attribute, which has motivated our research initiative. It is crucial to bridge the identified gap in the

```
00:00:00   -   10.16.120.44    -   298    -   legitimate
00:00:00   -   10.16.120.72    -   204    -   legitimate
00:00:00   -   192.168.0.150   -     2    -   legitimate
00:00:00   -   192.168.0.151   -     2    -   legitimate
00:00:00   -   192.168.0.152   -     2    -   legitimate
00:00:00   -   192.168.0.154   -     2    -   legitimate
00:00:00   -   192.168.0.173   -     2    -   legitimate
00:00:00   -   192.168.0.180   -     2    -   legitimate
00:00:00   -   192.168.1.100   -     4    -   DoS
00:00:00   -   192.168.1.190   -  3202    -   DoS
00:00:00   -   192.168.1.191   -  1762    -   legitimate
01:00:00   -   10.16.100.73    -    38    -   legitimate
01:00:00   -   10.16.120.44    -   236    -   legitimate
01:00:00   -   10.16.120.72    -   199    -   legitimate
01:00:00   -   192.168.0.150   -     4    -   legitimate
01:00:00   -   192.168.0.151   -     2    -   legitimate
01:00:00   -   192.168.0.152   -     2    -   legitimate
01:00:00   -   192.168.0.154   -     4    -   legitimate
01:00:00   -   192.168.0.155   -     6    -   legitimate
01:00:00   -   192.168.0.173   -     4    -   legitimate
01:00:00   -   192.168.0.174   -     6    -   legitimate
01:00:00   -   192.168.0.176   -     6    -   legitimate
01:00:00   -   192.168.0.178   -     4    -   legitimate
01:00:00   -   192.168.0.180   -     4    -   legitimate
01:00:00   -   192.168.1.190   -  2081    -   DoS
01:00:00   -   192.168.1.191   -  1320    -   DoS
02:00:00   -   10.16.120.44    -   181    -   legitimate
```

FIGURE 6. Extract attributes from the PCAP file.

```
    tcp.flags    Date   Time       Source          Length   Src_Port
0   0x00000002   01/01  12:00 AM   192.168.1.90        78      55363
1   0x00000002   01/01  12:37 AM   192.168.1.91        66       1883
2   0x00000002   01/01  12:38 AM   192.168.1.90        54      55363
3   0x00000002   01/01  12:38 AM   192.168.1.90        68      55363
4   0x00000002   01/01  12:40 AM   192.168.1.91        60       1883
```

FIGURE 7. Add 'source' attribute into MQTTset (Reduced) dataset.

TABLE 7. Attributes description.

Attributes	description
Date	month / date
Time	Hrs : mins : sec : AM / PM
Source	Source IP
Length	number of packet count
Src_Port	Source port
target	'legitimate' / 'DoS' attack

dataset through the strategic incorporation of 'source' as a key attribute. Notably, the attributes extracted from the PCAP file for this study include 'Date', 'Time', 'Source', 'Length,' 'Src_Port,' and 'target' Table 7 shows the attributes description.

This comprehensive approach seeks to enrich the dataset's analytical potential and contribute valuable insights to the fields of data analysis and cybersecurity.

D. AUTOMATED HYPERPARAMETER TUNING

Once the most effective model has been identified, its performance can be refined by fine-tuning its hyperparameters. It improves the overall security and reliability of the IoT-based system by providing early warning of potential threats. Hyperparameters are adjustable settings that determine how the ML algorithm will learn and perform. Fine-tuning of hyperparameters can be achieved by using grid search, random search, or Bayesian optimization techniques, among others. By adjusting the values of the model's hyperparameters, we can improve its accuracy and reduce its computational cost. This process involves systematically varying the values of each hyperparameter and evaluating the model's performance on a validation set.

The fine-tuning the hyperparameters of ML models can improve the performance of the model in terms of accuracy, precision, recall, and F1-score. It improves the reliability of the anomaly detection system and reduce the likelihood of false positives and false negatives. In summary, benchmarking analysis is important for evaluating the performance of different models, and fine-tuning the hyperparameters can further improve the accuracy and efficiency of the selected model. It can help to detect and mitigate potential threats and attacks in real-time and improve the overall security and reliability of the system.

E. EXPERIMENTAL DESIGN

The experiments were carried out in Colab [46]. Holdout validation was applied by separating the dataset into two groups, 70/30 split was applied with 70% of the data being used to train the model and the remaining 30% being used to test and evaluate it. This evaluation method was chosen to compare the results with the original benchmark MQTTset dataset.

IV. RESULTS OF FEATURE ANALYSIS

The next step which is related to the benchmarking analysis, it is important because it allows us to evaluate the performance of different models or algorithms in a standardized and objective manner. By comparing the results of different models on the same dataset using the same metrics, we can determine which approach is most effective for a particular task. In the context of developing an efficient ML approach for anomaly detection in the MQTTset dataset, benchmarking analysis can help to identify the most accurate and efficient model. By comparing the performance of different algorithms on the same dataset using appropriate metrics, we can determine which approach is most effective for detecting anomalies in the data.

A. EVALUATION OF BENCHMARK & VARIANTS OF DT

In the comprehensive evaluation of different DT benchmarks and their variations, extending its investigation to include a diverse range of ML algorithms. Assessing performance metrics within the realm of a prediction task, the analysis systematically examined the effects of both reduced and modified datasets on prediction accuracy and F1 score. This comprehensive approach involved an array of machine learning algorithms, allowing for a broader understanding of their efficiency. By placing decision trees alongside various algorithms in this extensive examination, the research aimed to

TABLE 8. Comparison of Prediction accuracies, F1 score of the benchmark algorithm – Before and after applying Feature Engineering.

Algos	Prediction Accuracy (%)		F1 Score (%)	
	Reduced DS	**Modify DS**	**Reduced DS**	**Modify DS**
NB	79.47%	80.17%	76.51%	77.96%
MLP	82.77%	94.06%	81.73%	92.72%
ANN	81.83%	85.10%	80.62%	83.09%
LR	79.71%	78.81%	77.02%	75.17%
DT	**92.57%**	**98.56%**	**92.38%**	**98.50%**
RF	92.56%	98.56%	92.38%	98.50%
DNN	82.72%	86.76%	81.60%	88.36%
KNN	91.32%	98.44%	91.23%	98.37%

offer a holistic perspective on their comparative effectiveness. The study served as a pivotal exploration into anomaly detection within IoT-based systems, utilizing the MQTTset dataset as a foundational element for the analysis.

Results: Benchmark Algorithms

Table 8 provided summarizes the performance metrics of different ML algorithms on a prediction task, evaluating both their reduced and modified datasets in terms of prediction accuracy and F1 score. The algorithms examined include NB, MLP, ANN, LR, DT, RF, DNN, and KNN. NB, it achieves reasonable accuracy and F1 Scores on both the Reduced and Modified datasets. While not the top performer, NB demonstrates consistency across datasets. MLP stands out with the highest accuracy and F1 Scores in the Modified DS. This indicates its proficiency in capturing complex patterns and nuances present in the data, making it a strong contender for scenarios demanding precision and recall. ANN shows competitive results, particularly in the Modified DS. Its ability to adapt to intricate relationships in data is reflected in the higher F1 Score, making it a reliable choice for tasks requiring nuanced predictions. LR, while not leading in terms of metrics, maintains a steady performance across both datasets. It may be considered a pragmatic choice for simpler models or when interpretability is crucial. DT and RF shine in both accuracy and F1 Score, especially in the Modified DS. These tree-based models demonstrate robustness and effectiveness in capturing complex decision boundaries. DNN performs well, showcasing an improvement in F1 Score in the Modified DS. DNNs are known for their capacity to model intricate relationships, and this is reflected in the enhanced performance in the more challenging dataset. KNN exhibits high accuracy and F1 Score, particularly excelling in the Modified DS. KNN's strength lies in capturing local patterns, making it effective in scenarios where instances with similar features tend to belong to the same class.

Figure 9 shows, NB exhibits consistent performance across both datasets, showcasing moderate accuracy of 79.47% in the Reduced DS and a slightly improved 80.17% in the Modified DS. Correspondingly, the F1 Scores also demonstrate stability, with values of 76.51% and 77.96% in the Reduced and Modified DS, respectively. MLP emerges as a standout performer, particularly excelling in the Modified DS. With an accuracy of 82.77% in the Reduced DS and a remarkable 94.06% in the Modified DS, coupled with F1 Scores of 81.73% and 92.72%, MLP demonstrates its capability to capture complex patterns and nuances in the data. ANN delivers solid performance, especially in the Modified DS. With an accuracy of 81.83% in the Reduced DS and an improved 85.10% in the Modified DS, alongside F1 Scores of 80.62% and 83.09%, ANN showcases its adaptability to intricate relationships in the data. LR maintains a steady performance, showing similar accuracy and F1 Scores across both datasets. With an accuracy of 79.71% in the Reduced DS and 78.81% in the Modified DS, LR demonstrates reliability in simpler models or scenarios where interpretability is crucial.

DT and RF emerge as standout performers, particularly excelling in the Modified DS. DT achieves an accuracy of 92.57% in the Reduced DS and an exceptional 98.56% in the Modified DS, with corresponding F1 Scores of 92.38% and 98.50%. RF mirrors DT's exceptional performance, showcasing robust accuracy and F1 Scores, especially in the Modified DS. DNN demonstrates competitive performance, with an accuracy of 82.72% in the Reduced DS and an improved 86.76% in the Modified DS. Notable improvements in F1 Score are observed in the Modified DS, reaching 88.36%. KNN exhibits high accuracy and F1 Scores, particularly excelling in the Modified DS. With an accuracy of 91.32% in the Reduced DS and an outstanding 98.44% in the Modified DS, coupled with F1 Scores of 91.23% and 98.37%, KNN proves effective in capturing local patterns and achieving precision-recall balance.

Similarly, in terms of F1 score, MLP stood out with an F1 score of 92.72%, closely followed by RF and DT at 98.50% see Figure 10.

Overall, the results suggest that MLP, RF, and DT are strong performers across both the reduced and modified datasets, showcasing their robustness in this predictive task. However, it's essential to consider other factors such as computational complexity and interpretability when selecting the most suitable algorithm for a specific application.

Table 9 provided offers insights into the training and testing times of various ML algorithms when applied to both a reduced and modified dataset. These times are measured in seconds (sec).

For training time on the reduced dataset, the algorithms exhibit a wide range of performance. KNN stands out as the fastest, with a mere 0.41 seconds required for training, followed by DT at 2.95 seconds. In contrast, MLP consumes the most time, with a considerable 338.56 seconds needed for training. Other algorithms such as ANN and DNN also

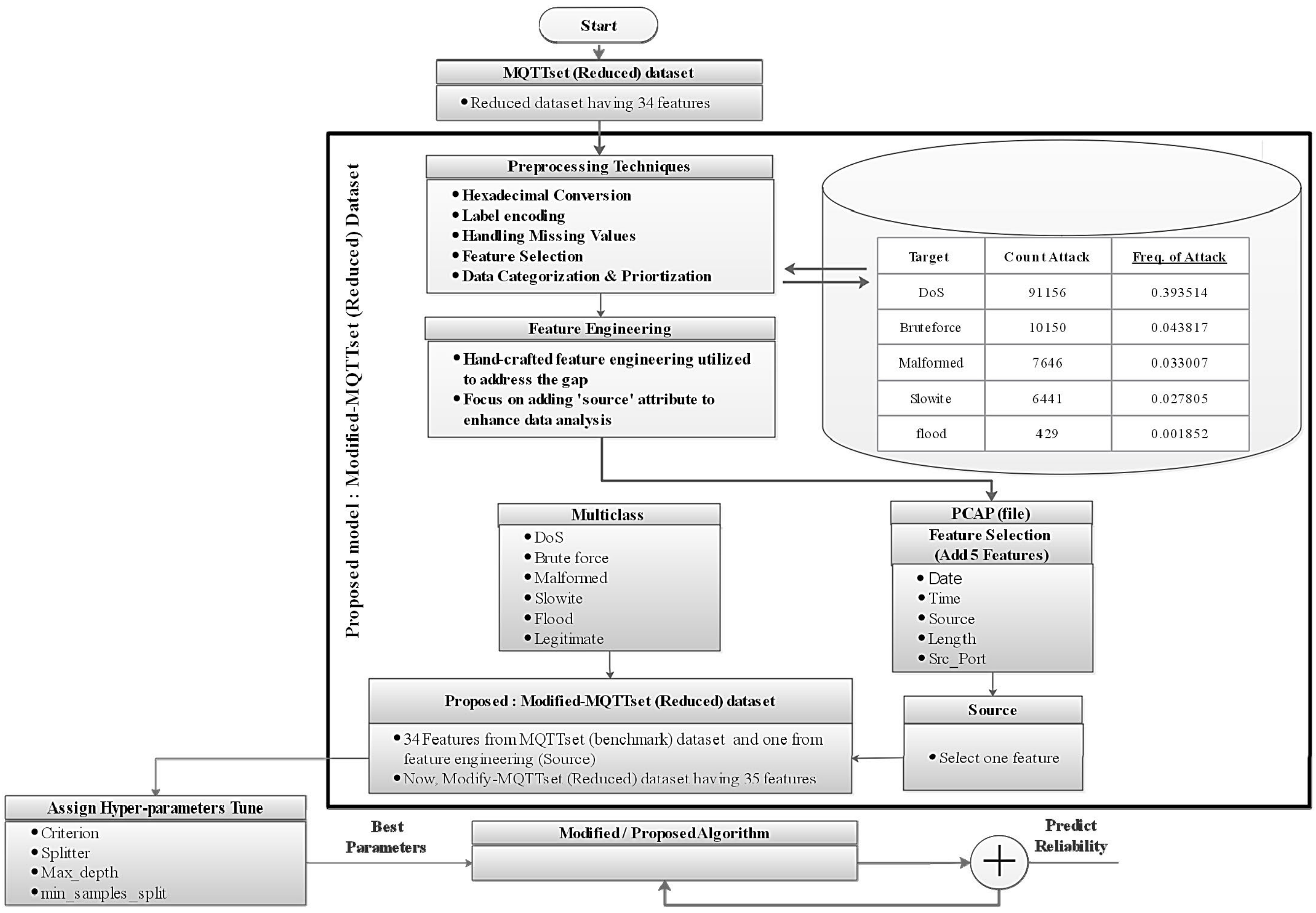

Start
MQTTset (Reduced) dataset
• Reduced dataset having 34 features
Proposed model : Modified-MQTTset (Reduced) Dataset
Preprocessing Techniques
• Hexadecimal Conversion
• Label encoding
• Handling Missing Values
• Feature Selection
• Data Categorization & Priortization
Target | Count Attack | Freq. of Attack
DoS | 91156 | 0.393514
Bruteforce | 10150 | 0.043817
Malformed | 7646 | 0.033007
Slowite | 6441 | 0.027805
flood | 429 | 0.001852
Feature Engineering
• Hand-crafted feature engineering utilized to address the gap
• Focus on adding 'source' attribute to enhance data analysis
Multiclass
• DoS
• Brute force
• Malformed
• Slowite
• Flood
• Legitimate
PCAP (file)
Feature Selection (Add 5 Features)
• Date
• Time
• Source
• Length
• Src_Port
Proposed : Modified-MQTTset (Reduced) dataset
• 34 Features from MQTTset (benchmark) dataset and one from feature engineering (Source)
• Now, Modify-MQTTset (Reduced) dataset having 35 features
Source
• Select one feature
Assign Hyper-parameters Tune
• Criterion
• Splitter
• Max_depth
• min_samples_split
Best Parameters
Modified / Proposed Algorithm
Predict Reliability

FIGURE 8. Proposed model: Modified MQTTset (Reduced) method of MQTTset dataset.

TABLE 9. Comparison of computational training and testing time of the benchmark algorithm – Before and after applying Feature Engineering.

Algos	Training Time (sec)		Testing Time (sec)	
	Reduced DS	**Modify DS**	**Reduced DS**	**Modify DS**
NB	1.39	2.08	1.08	1.13
MLP	338.56	317.12	0.57	0.75
ANN	159.49	145.40	10.42	10.45
LR	30.70	20.30	0.14	0.12
DT	**2.95**	**0.70**	**0.86**	**0.02**
RF	15.13	12.93	1.42	1.03
DNN	202.58	202.64	6.08	6.28
KNN	0.41	1.01	548.42	165.55

demand substantial training times, approximately 159.49 seconds and 202.58 seconds, respectively.

When transitioning to the modified dataset, the training times generally decrease for most algorithms. Notably, MLP's training time drops to 317.12 seconds, while ANN's training time is reduced to 145.40 seconds. KNN remains the fastest for training on the modified dataset, taking only 1.01 seconds.

Turning to testing times on the reduced dataset, most algorithms perform relatively quickly. LR and DT have the fastest testing times at 0.14 seconds and 0.86 seconds, respectively.

On the other hand, KNN's testing time significantly increases to 548.42 seconds, suggesting it may not be the best choice for scenarios requiring speedy predictions on this dataset.

In the case of the modified dataset, testing times remain within reasonable limits for all algorithms, with the longest time being 10.45 seconds for ANN. These times are generally acceptable for most applications, but considerations should be made depending on real-time requirem ents and computational resources.

Results: DT and its Variants

Table 10 presents the performance metrics of nine different DT and its variant algorithms on two distinct datasets: a reduced dataset and a modified dataset. These algorithms are evaluated based on their prediction accuracy and F1 score, which are essential metrics for assessing classification models.

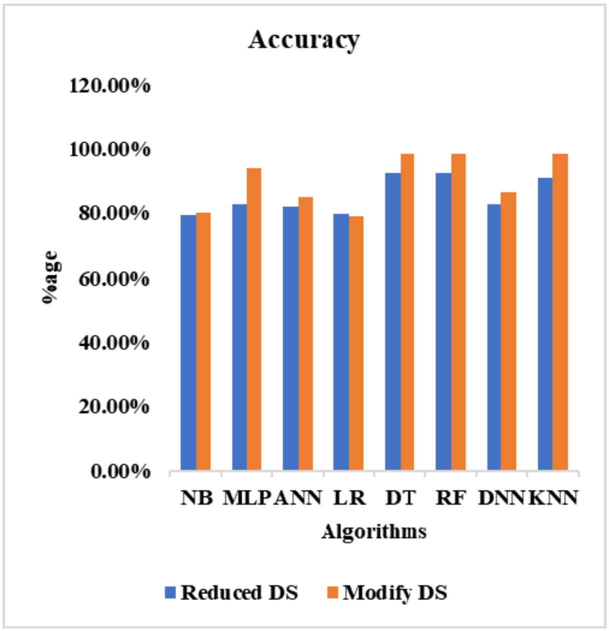


FIGURE 9. Comparison chart of Prediction accuracies of the benchmark algorithm – Before and after applying Feature Engineering.

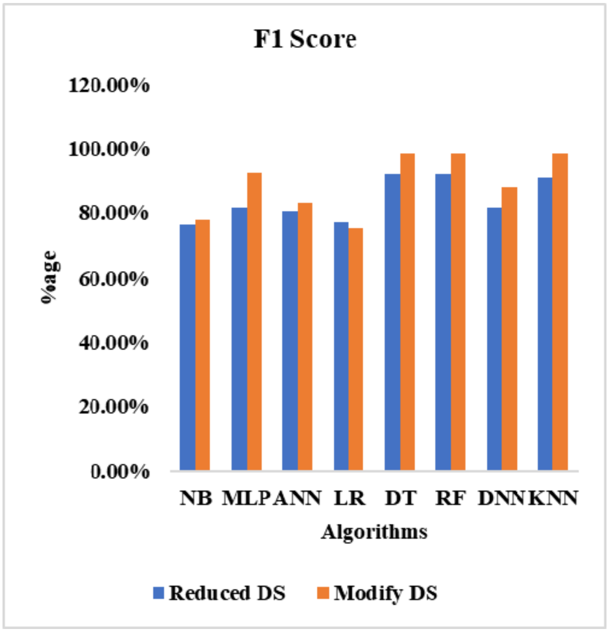


FIGURE 10. Comparison chart of F1 score of the benchmark algorithm – Before and after applying Feature Engineering.

The DT, RF, ID3, CatBoost, LightGBM, XGBoost, C4.5, GB, and CART models all exhibit high prediction accuracy ranging from 92.54% to 92.80%. This indicates that these algorithms are generally effective in making correct predictions on the dataset. The high F1 scores, ranging from 98.49% to 98.56%, further validate the models' ability to strike a balance between precision and recall. CatBoost and LightGBM outperform the other models with the highest prediction accuracy and F1 scores, demonstrating their robustness on the MQTTset dataset. XGBoost, RF, and CART also perform consistently well across both metrics see Figure 13. This demonstrates their robustness and reliability in classification tasks, regardless of the dataset variation.

TABLE 10. Comparison of Prediction accuracies, F1 score of the DT and its Variants – Before and after applying Feature Engineering.

Algos	Prediction Accuracy (%)		F1 Score (%)	
	Reduced DS	Modify DS	Reduced DS	Modify DS
DT	**92.57%**	**98.56%**	**92.38%**	**98.50%**
RF	92.56%	98.56%	92.38%	98.50%
ID3	92.55%	98.55%	92.37%	98.49%
CatBoost	92.80%	98.55%	92.62%	98.49%
LightGBM	92.80%	98.49%	92.63%	98.49%
XGBoost	92.54%	98.56%	92.36%	98.50%
C4.5	92.56%	98.55%	92.37%	98.49%
GB	92.55%	98.55%	92.33%	98.49%
CART	92.56%	98.56%	92.38%	98.50%

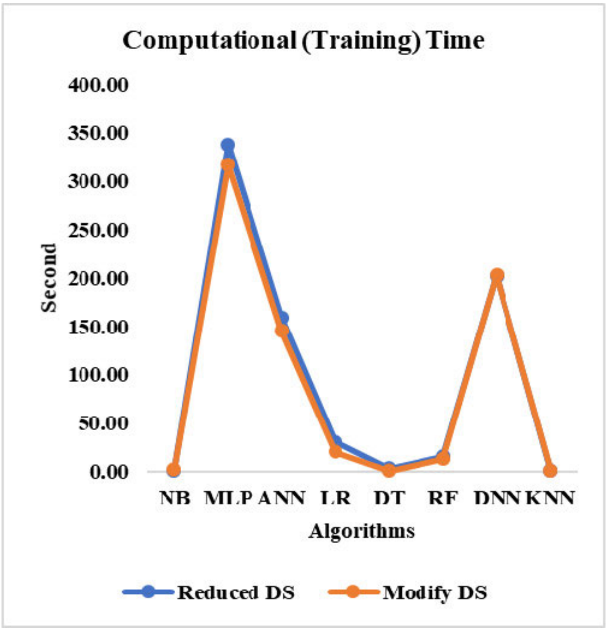


FIGURE 11. Comparison of computational training time of the benchmark algorithm – Before and after applying Feature Engineering.

Similarly, when considering the F1 score, these algorithms maintain remarkably consistent performance see Figure 14, with scores ranging from 92.33% to 92.63% on the reduced dataset and 98.49% to 98.50% on the modified dataset. This consistent performance across both datasets underscores the reliability of these DT and variant algorithms in maintaining high precision and recall for classification tasks.

Table 11 provides insights into the computational performance of nine DT and its variant algorithms in terms of training and testing times, measured in seconds (sec). These algorithms have been evaluated on both a reduced dataset and a modified dataset. In terms of training time, the fastest models in terms of training time are DT, ID3, and CART. C4.5 and RF have moderate training times. GB, CatBoost,

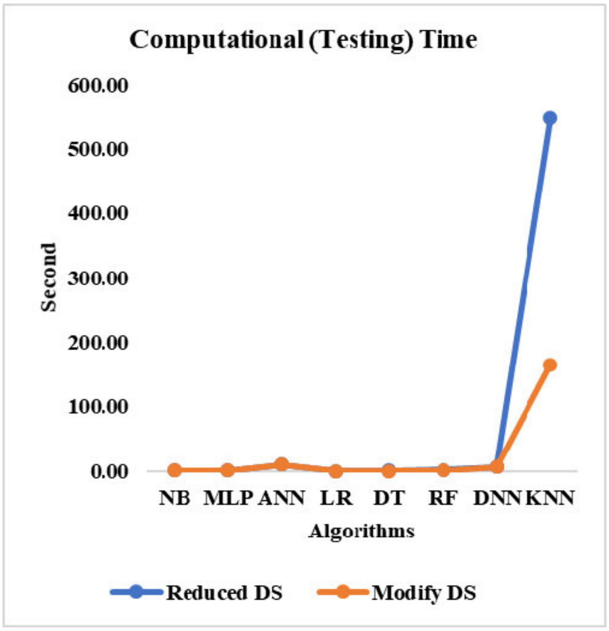


FIGURE 12. Comparison of computational testing time of the benchmark algorithm – Before and after applying Feature Engineering.

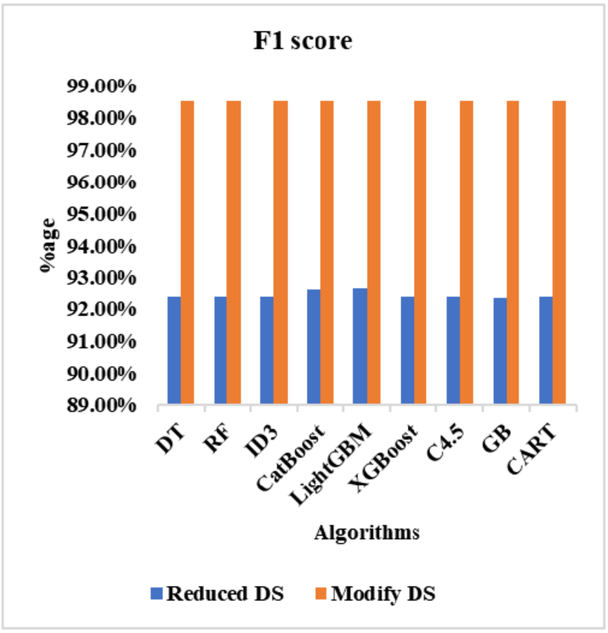


FIGURE 14. Comparison chart of F1 score of the DT and Its Variants – Before and after applying Feature Engineering.

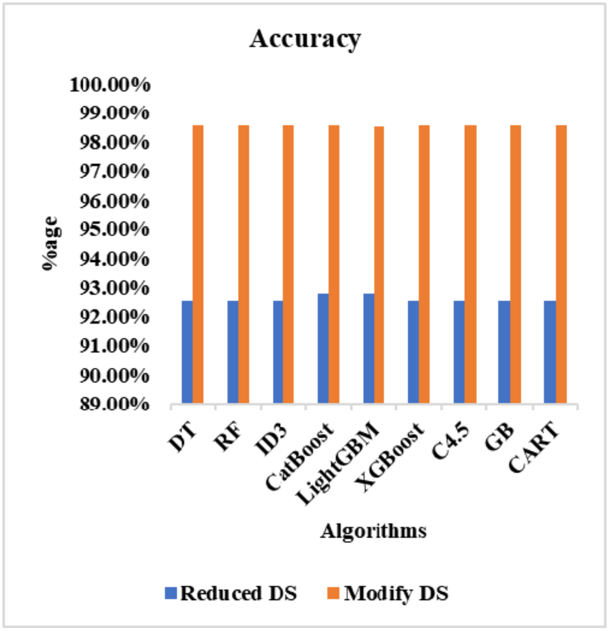


FIGURE 13. Comparison chart of Prediction accuracies of the DT and Its Variants – Before and after applying Feature Engineering.

TABLE 11. Comparison of computational training and testing time of the DT and its Variants – Before and after applying Feature Engineering.

Algos	Training Time (sec)		Testing Time (sec)	
	Reduced DS	Modify DS	Reduced DS	Modify DS
DT	**2.95**	**0.70**	**0.86**	**0.02**
RF	15.13	12.93	1.42	1.03
ID3	3.11	1.21	0.83	0.43
CatBoost	43.12	49.87	0.87	0.29
LightGBM	20.93	14.25	4.50	2.41
XGBoost	14.65	11.29	1.37	0.97
C4.5	4.37	1.22	0.40	0.02
GB	206.38	233.40	1.54	1.77
CART	1.38	1.28	0.23	0.02

LightGBM, and XGBoost have significantly longer training times. In testing time, the DT, ID3, and CART are the fastest models. CatBoost, XGBoost, and LightGBM have higher testing times, with CatBoost having the lowest among them. GB has the longest testing time. The overall performance of DT, ID3, and CART seems to provide a good balance between training and testing times. RF shows a longer training time but potentially better performance with its ensemble approach. CatBoost, LightGBM, and XGBoost are boosting algorithms with longer training times but may offer better predictive performance due to their ensemble nature. GB has the longest training time, which might be a concern in scenarios where speed is crucial. The potential considerations, if model interpretability and fast predictions are essential, are DT, ID3, and CART. If a trade-off between training time and potential accuracy is acceptable, Random Forest (RF) might be a good choice. For datasets where boosting algorithms tend to perform well, CatBoost, LightGBM, and XGBoost could be suitable, but their longer training times need to be considered.

Figure 15 shows in terms of training times on the reduced dataset, the algorithms exhibit notable variations. DT and ID3 are the fastest, requiring just 2.95 seconds and 3.11 seconds,

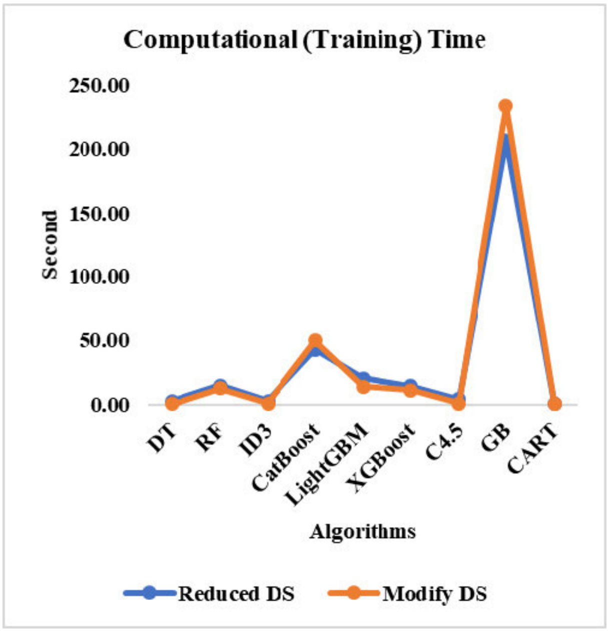


FIGURE 15. Comparison chart of computational training time of the DT and Its Variants – Before and after applying Feature Engineering.

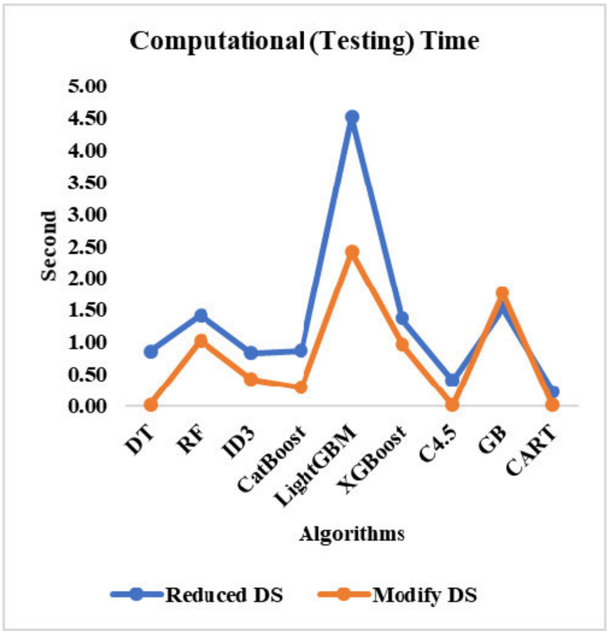


FIGURE 16. Comparison chart of computational testing time of the DT and Its Variants – Before and after applying Feature Engineering.

respectively, for training. On the other hand, GB is the most time-consuming, taking a substantial 206.38 seconds. This variance in training times suggests that certain DT variants are more computationally efficient for this specific dataset.

Transitioning to the modified dataset, training times generally increase, but the relative rankings remain consistent.

DT and ID3 maintain their efficiency, with minimal increases in training times. Conversely, GB remains the slowest but exhibits a more significant increase in training time compared to the other algorithms.

For testing times on the reduced dataset, most algorithms perform quite swiftly, with C4.5 being the fastest at just 0.40 seconds see Figure 16. However, on the modified dataset, there are slight increases in testing times for most algorithms, yet they generally remain within reasonable limits.

In summary, this table illustrates the varying computational demands of nine DT and its variant algorithms on both the reduced and modified datasets. While certain algorithms like DT and ID3 excel in terms of training efficiency, others like GB are more computationally intensive but may offer improved predictive performance. The choice of algorithm should be carefully considered, considering both computational resources and the desired predictive accuracy for a given dataset and application.

Results: Best Automated parameter tuning in DT

Table 12 provided to observe the parameter grid for fine-tuning a DT algorithm, focusing on the criteria for splitting and the splitter selection method. The goal is to identify the best automated parameters for optimizing the DT model's performance.

In terms of the CRITERION parameter, which determines the criterion used for splitting nodes, the table presents two options: Gini and Entropy. The AUTO-TUNED PARAMETERS column indicates that the best automated parameter for this criterion is Gini. This suggests that, in this specific context, using the Gini impurity as the criterion for node splitting yields better results than using the Entropy criterion.

TABLE 12. Best Automated parameter tuning w.r.t criterion and splitter.

Parameter grid	parameter		Auto-tuned Parameters
criterion	Gini	Entropy	**Gini**
splitter	Best	Random	**Best**

The SPLITTER parameter, which specifies the strategy for choosing the split at each node, the table offers two choices: Best and Random. According to the AUTO-TUNED PARAMETERS column, the best automated parameter for this setting is "Best." This implies that selecting the best possible split at each node, rather than relying on random splits, is the preferred strategy for optimizing the DT model's performance.

The automated parameter tuning results, the best settings for the DT algorithm in this context are to use the Gini criterion for splitting nodes and the "Best" strategy for selecting the split. These parameter choices have been determined to provide the most favorable outcomes for this specific DT model and dataset. However, it's essential to note that parameter tuning can be dataset-dependent, and the best settings may vary for different datasets and problem domains.

Table 13 provided outlines the parameter grid for fine-tuning a DT algorithm, with a focus on the maximum depth of the tree, the minimum number of samples required to split an internal node (MIN_SAMPLES_SPLIT), and

TABLE 13. Best Automated parameter tuning w.r.t maximum depth, minimum samples split and leaf.

Parameter grid	parameter			Auto-tuned Parameters
Max depth	None	5	10	**None**
Min_samples_split	2	5	10	**10**
Min_samples_leaf	1	2	4	**1**

TABLE 14. Modify DT Algorithms using hyper parameter tune.

Algorithms	Accuracy (%)	F1 score (%)	Training Time (Sec)	Testing Time (Sec)
Modify DT Algorithms	99.27%	99.26%	0.73	0.14

the minimum number of samples required to be at a leaf node (MIN_SAMPLES_LEAF). The goal is to identify the best automated parameters for optimizing the DT model's performance.

Starting with the MAX DEPTH parameter, the DEFAULT PARAMETER column indicates that the default value is set to "None," which means the tree can grow without any restrictions on its depth. However, through auto-tuning, the best automated parameter for this setting is determined to be 5, suggesting that constraining the maximum depth of the tree to 5 levels yields improved performance.

The MIN_SAMPLES_SPLIT parameter, the DEFAULT PARAMETER shows that the default value is set to 2, indicating that a node can be split as long as it contains at least 2 samples. However, the AUTO-TUNED PARAMETERS column reveals that, through automated tuning, the best parameter setting is 10. This suggests that requiring a minimum of 10 samples to split an internal node is more favorable for optimizing the DT's performance in this context.

The MIN_SAMPLES_LEAF parameter, the DEFAULT PARAMETER is set to 1, implying that a leaf node can contain just one sample. However, the AUTO-TUNED PARAMETERS column indicates that the best automated parameter for this setting is 1, which means keeping the minimum samples per leaf at its default value is the most suitable choice for this DT model.

In summary, based on the automated parameter tuning results, the best parameter settings for the DT algorithm in this particular context are to limit the maximum depth of the tree to 5 levels, require a minimum of 10 samples to split an internal node, and keep the minimum samples per leaf at its default value of 1. These parameter choices are determined to provide the most favorable outcomes for this specific DT model and dataset.

Table 14 provides an overview of the performance metrics for modified DT algorithms. The accuracy of these algorithms stands impressively high at 99.27%, indicating their proficiency in correctly classifying data points during training.

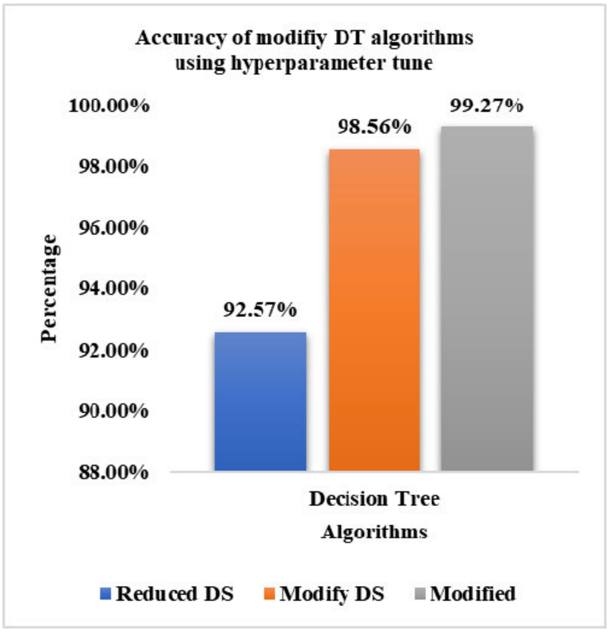


FIGURE 17. Accuracy of modify DT algorithms using hyperparameter tune.

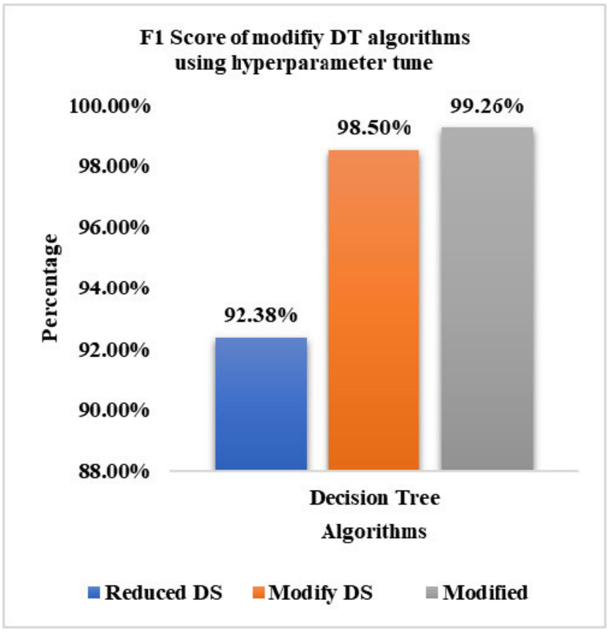


FIGURE 18. F1 score of modify DT algorithms using hyper parameter tune.

Additionally, the F1 score, a measure of the algorithms' precision and recall, is equally impressive at 99.26%, signifying their ability to balance between correctly identifying positive cases and minimizing false positives.

Moreover, the training time required for these modified DT algorithms is remarkably low, clocking in at just 0.73 seconds, showcasing their efficiency in model development. When it comes to testing, these algorithms maintain a swift pace with a testing time of only 0.14 seconds, ensuring quick predictions on new data. Overall, these modified DT algorithms exhibit outstanding performance across various

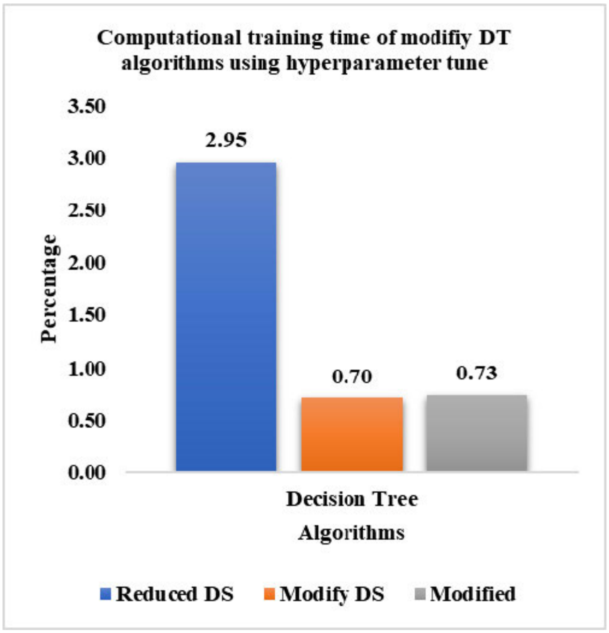


FIGURE 19. Computational training time of modify DT algorithms using hyper parameter tune.

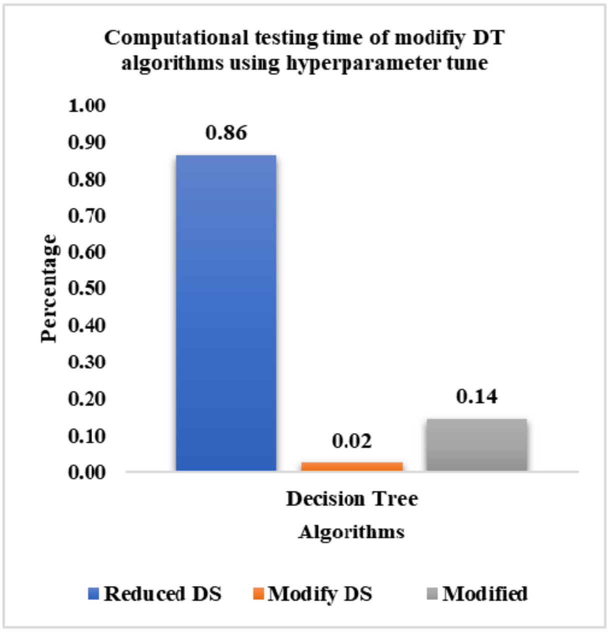


FIGURE 20. Computational testing time of modify DT algorithms using hyper parameter tune.

metrics, making them a valuable asset in data classification tasks.

The comprehensive comparison of accuracy, F1 scores, training time, and testing time for DT algorithms, including the original DT, Reduced DT, and Modified DT algorithms. In terms of accuracy, the Modified DT Algorithm outperforms the other two, Figure 17 shows, achieving an impressive accuracy rate of 99.27%, showcasing its ability to make precise predictions. Similarly, Figure 18 shows the F1 score for the Modified DT Algorithm stands at 99.26%, indicating a remarkable balance between precision and recall.

When it comes to efficiency, the training time for the Modified DT Algorithm is notably shorter at 0.73 seconds, making it a more time-efficient choice for model development compared to the original DT (2.95 seconds) and Reduced DT (0.70 seconds).

In terms of testing time, the Modified DT Algorithm is the quickest, taking only 0.14 seconds for predictions, whereas the Reduced DT takes 0.02 seconds, and the original DT takes 0.86 seconds.

Overall, these results suggest that the Modified DT Algorithm offers a compelling combination of high accuracy, F1 score, and efficiency, making it a strong candidate for various data classification tasks, especially when speed and precision are essential.

V. CONCLUSION

In conclusion, the research conducted on the MQTTset dataset represents a significant advancement in the field of anomaly detection within IoT-based systems. This study addressed a critical gap in the existing literature by incorporating the 'source' attribute from PCAP files and employing hand-crafted feature engineering techniques. The primary objective was to enhance the identification of anomalies, particularly focusing on the detection of DoS attacks.

The evaluation of various ML models, including the benchmark DT and its eight variant models, demonstrated the effectiveness of feature engineering in improving detection accuracy and F1 scores. The results showcased substantial performance gains, with the Modified DT Algorithm achieving an impressive 99.27% accuracy and a 99.26% F1 score. Notably, this improvement was accompanied by a reduction in both training and testing times, enhancing computational efficiency.

Furthermore, the incorporation of hyperparameter fine-tuning techniques, such as grid search and random search, further refined the model performance, ensuring higher accuracy while managing computational costs.

Overall, this research contributes valuable insights into the realm of feature engineering and offers clear guidance on selecting the most effective approach for anomaly detection in IoT-based systems. The proposed method not only provides early threat warnings but also significantly enhances the overall security and reliability of IoT-based systems, making it a promising avenue for future research and practical implementation. In conclusion, the future directions for this research involve delving deeper into advanced techniques, scalability, adaptability, real-world implementation, and continuous improvement to meet the evolving landscape of IoT-based systems and their security challenges.

REFERENCES

[1] C. Dziuban, P. Moskal, and J. Hartman, "Higher education, blended learning, and the generations: Knowledge is power-no more," in *Elements of Quality Online Education: Engaging Communities*. Needham, MA, USA: Sloan Center for Online Education, 2005.

[2] S. Holden. *Computer Laboratory, Room FC06 Telephone Extension 63725 Email: Sbh11@cam.ac.uk*. [Online]. Available: http://www.cl.cam.ac.uk/users/sbh11/

[3] D. Bandyopadhyay and J. Sen, "Internet of Things: Applications and challenges in technology and standardization," 2011, *arXiv:1105.1693*.

[4] E. F. Ogbomo, "Importance of information and communication technologies (ICTs) in making a heathy information society: A case study of ethiope east local government area of delta state, Nigeria," *Library Philosophy Pract.*, vol. 3, no. 1, pp. 1–8, 2008.

[5] O. Ali, M. K. Ishak, M. K. L. Bhatti, I. Khan, and K.-I. Kim, "A comprehensive review of Internet of Things: Technology stack, middlewares, and fog/edge computing interface," *Sensors*, vol. 22, no. 3, p. 995, Jan. 2022, doi: 10.3390/s22030995.

[6] M. Ahad, G. Tripathi, S. Zafar, and F. Doja, "IoT data management—Security aspects of information linkage in IoT systems," in *Principles of Internet of Things (IoT) Ecosystem: Insight Paradigm*, 2020, pp. 439–464, doi: 10.1007/978-3-030-33596-0_18.

[7] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Gener. Comput. Syst.*, vol. 29, no. 27, pp. 1645–1660, 2013, doi: 10.1016/j.future.2013.01.010.

[8] M. Zolanvari, M. A. Teixeira, L. Gupta, K. M. Khan, and R. Jain, "Machine learning-based network vulnerability analysis of industrial Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 6822–6834, Aug. 2019, doi: 10.1109/JIOT.2019.2912022.

[9] M. Chernyshev, Z. Baig, O. Bello, and S. Zeadally, "Internet of Things (IoT): Research, simulators, and testbeds," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 1637–1647, Jun. 2018, doi: 10.1109/JIOT.2017.2786639.

[10] A. Guezzaz, A. Asimi, Y. Sadqi, Y. Asimi, and Z. Tbatou, "A new hybrid network sniffer model based on pcap language and sockets (Pcapsocks)," *Int. J. Adv. Comput. Sci. Appl.*, vol. 7, no. 2, pp. 1–10, Feb. 2016, doi: 10.14569/ijacsa.2016.070228.

[11] S. Saad, I. Traore, A. Ghorbani, B. Sayed, D. Zhao, W. Lu, J. Felix, and P. Hakimian, "Detecting P2P botnets through network behavior analysis and machine learning," in *Proc. 9th Annu. Int. Conf. Privacy, Secur. Trust*, Jul. 2011, pp. 174–180, doi: 10.1109/PST.2011.5971980.

[12] R. H. Bajaj and P. L. Ramteke, "Big data—The new era of data," Tech. Rep., 2014, vol. 5.

[13] *Sensors | Free Full-Text | Simulation of Attacks for Security in Wireless Sensor Network*. Accessed: Jan. 3, 2024. [Online]. Available: https://www.mdpi.com/1424-8220/16/11/1932

[14] Z. Li, Y. Zhu, and M. Van Leeuwen, "A survey on explainable anomaly detection," *ACM Trans. Knowl. Discovery Data*, vol. 18, no. 1, pp. 1–54, Jan. 2024, doi: 10.1145/3609333.

[15] A. Diro, N. Chilamkurti, V.-D. Nguyen, and W. Heyne, "A comprehensive study of anomaly detection schemes in IoT networks using machine learning algorithms," *Sensors*, vol. 21, no. 24, p. 8320, Dec. 2021, doi: 10.3390/s21248320.

[16] M. Fahim and A. Sillitti, "Anomaly detection, analysis and prediction techniques in IoT environment: A systematic literature review," *IEEE Access*, vol. 7, pp. 81664–81681, 2019, doi: 10.1109/ACCESS.2019.2921912.

[17] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection for discrete sequences: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 5, pp. 823–839, May 2012, doi: 10.1109/TKDE.2010.235.

[18] N. Naik. *MQTT: The Standard for IoT Messaging*. [Online]. Available: https://mqtt.org/

[19] 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, 2020, doi: 10.3390/s20226578.

[20] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani. *A Detailed Analysis of the CICIDS2017 Data Set | SpringerLink*. Accessed: Aug. 31, 2023. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-25109-3_9

[21] F. Abbasi, M. Naderan, and S. E. Alavi, "Anomaly detection in Internet of Things using feature selection and classification based on logistic regression and artificial neural network on N-BaIoT dataset," in *Proc. 5th Int. Conf. Internet Things Appl. (IoT)*, 2021, pp. 1–7, doi: 10.1109/IoT52625.2021.9469605.

[22] Z. Gu, S. Nazir, C. Hong, and S. Khan, "Convolution neural network-based higher accurate intrusion identification system for the network security and communication," *Secur. Commun. Netw.*, vol. 2020, pp. 1–10, Aug. 2020. [Online]. Available: https://www.hindawi.com/journals/scn/2020/8830903/

[23] B. Kaur, S. Dadkhah, F. Shoeleh, E. C. P. Neto, P. Xiong, S. Iqbal, P. Lamontagne, S. Ray, and A. A. Ghorbani, "Internet of Things (IoT) security dataset evolution: Challenges and future directions," *Internet Things*, vol. 22, Jul. 2023, Art. no. 100780. https://www.sciencedirect.com/science/article/abs/pii/S2542660523001038

[24] N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac, and P. Faruki, "Network intrusion detection for IoT security based on learning techniques," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2671–2701, 3rd Quart., 2019, doi: 10.1109/COMST.2019.2896380.

[25] J. M. Peterson, J. L. Leevy, and T. M. Khoshgoftaar, "A review and analysis of the bot-IoT dataset," in *Proc. IEEE Int. Conf. Service-Oriented Syst. Eng. (SOSE)*, Aug. 2021, pp. 20–27, doi: 10.1109/SOSE52839.2021.00007.

[26] N. Abdalgawad, A. Sajun, Y. Kaddoura, I. A. Zualkernan, and F. Aloul. *Generative Deep Learning to Detect Cyberattacks for the IoT-23 Dataset | IEEE Journals & Magazine | IEEE Xplore*. Accessed: Aug. 31, 2023. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9667357

[27] F. Hussain, S. G. Abbas, M. Husnain, U. U. Fayyaz, F. Shahzad, and G. A. Shah, "IoT DoS and DDoS attack detection using ResNet," in *Proc. IEEE 23rd Int. Multitopic Conf. (INMIC)*, Nov. 2020, pp. 1–6, doi: 10.1109/INMIC50486.2020.9318216.

[28] H. Hindy, E. Bayne, M. Bures, and R. Atkinson. *Machine Learning Based IoT Intrusion Detection System: An MQTT Case Study (MQTT-IoT-IDS2020 Dataset) | SpringerLink*. Accessed: Aug. 31, 2023. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-64758-2_6

[29] *Sensors | Free Full-Text | MQTTset, a New Dataset for Machine Learning Techniques on MQTT*. Accessed: May 10, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/20/22/6578

[30] S. Choi and J. Cho, "Novel feature extraction method for detecting malicious MQTT traffic using Seq2Seq," *Appl. Sci.*, vol. 12, no. 23, p. 12306, Dec. 2022, doi: 10.3390/app122312306.

[31] C. Kingsford and S. L. Salzberg, "What are decision trees?" *Nature Biotechnol.*, vol. 26, no. 9, pp. 1011–1013, Sep. 2008, doi: 10.1038/nbt0908-1011.

[32] T. K. Ho, "Random decision forests," in *Proc. 3rd Int. Conf. Document Anal. Recognit.*, vol. 1, Aug. 1995, pp. 278–282, doi: 10.1109/ICDAR.1995.598994.

[33] N. Elmrabit, F. Zhou, F. Li, and H. Zhou, "Evaluation of machine learning algorithms for anomaly detection," in *Proc. Int. Conf. Cyber Secur. Protection Digit. Services*, Jun. 2020, pp. 1–8, doi: 10.1109/CyberSecurity49315.2020.9138871.

[34] M. Karabatak, "A new classifier for breast cancer detection based on Naïve Bayesian," *Measurement*, vol. 72, pp. 32–36, Aug. 2015, doi: 10.1016/j.measurement.2015.04.028.

[35] S. Dreiseitl and L. Ohno-Machado, "Logistic regression and artificial neural network classification models: A methodology review," *J. Biomed. Inform.*, vol. 35, nos. 5–6, pp. 352–359, 2002, doi: 10.1016/S1532-0464(03)00034-0.

[36] M. Madhiarasan and S. N. Deepa. *Comparative Analysis on Hidden Neurons Estimation in Multi Layer Perceptron Neural Networks for Wind Speed Forecasting | SpringerLink*. Accessed: Aug. 31, 2023. [Online]. Available: https://link.springer.com/article/10.1007/s10462-016-9506-6

[37] A. April, J. Basu, D. Bhattacharyya, and T.-H. Kim, "Use of artificial neural network in pattern recognition," *Int. J. Softw. Eng. Appl.*, vol. 4, no. 2, pp. 1–12, May 2020.

[38] T. Hossen, S. J. Plathottam, R. K. Angamuthu, P. Ranganathan, and H. Salehfar, "Short-term load forecasting using deep neural networks (DNN)," in *Proc. North Amer. Power Symp. (NAPS)*, Sep. 2017, pp. 1–6, doi: 10.1109/NAPS.2017.8107271.

[39] R. Shyam and R. Chakraborty, "Machine learning and its dominant paradigms," *J. Advancement Robot.*, vol. 8, no. 2, pp. 1–10, Sep. 2021.

[40] A. Navada, A. N. Ansari, S. Patil, and B. A. Sonkamble, "Overview of use of decision tree algorithms in machine learning," in *Proc. IEEE Control Syst. Graduate Res. Colloq.*, Jun. 2011, pp. 37–42, doi: 10.1109/ICSGRC.2011.5991826.

[41] B. Hssina, A. Merbouha, H. Ezzikouri, and M. Erritali, "A comparative study of decision tree ID3 and C4.5," *Int. J. Adv. Comput. Sci. Appl.*, vol. 4, no. 2, 2014, doi: 10.14569/specialissue.2014.040203.

[42] Z. Feng, H. Xiong, C. Song, S. Yang, B. Zhao, L. Wang, Z. Chen, S. Yang, L. Liu, and J. Huan, "SecureGBM: Secure multi-party gradient boosting," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2019, pp. 1312–1321, doi: 10.1109/BigData47090.2019.9006000.

[43] A. Gajjar, P. Kashyap, A. Aysu, P. Franzon, S. Dey, and C. Cheng, "FAXID: FPGA-accelerated XGBoost inference for data centers using HLS," in *Proc. IEEE 30th Annu. Int. Symp. Field-Programmable Custom Comput. Mach. (FCCM)*, May 2022, pp. 1–9, doi: 10.1109/FCCM53951.2022.9786085.

[44] S. B. Jabeur, C. Gharib, S. Mefteh-Wali, and W. B. Arfi, "CatBoost model and artificial intelligence techniques for corporate failure prediction," *Technol. Forecasting Social Change*, vol. 166, May 2021, Art. no. 120658, doi: 10.1016/j.techfore.2021.120658.

[45] B. Ma, F. Meng, G. Yan, H. Yan, B. Chai, and F. Song, "Diagnostic classification of cancers using extreme gradient boosting algorithm and multi-omics data," *Comput. Biol. Med.*, vol. 121, Jun. 2020, Art. no. 103761, doi: 10.1016/j.compbiomed.2020.103761.

[46] T. S. Gunawan, A. Ashraf, B. S. Riza, E. V. Haryanto, R. Rosnelly, M. Kartiwi, and Z. Janin, "Development of video-based emotion recognition using deep learning with Google Colab," *TELKOMNIKA, Telecommun. Comput. Electron. Control*, vol. 18, no. 5, p. 2463, Oct. 2020, doi: 10.12928/telkomnika.v18i5.16717.

IMRAN received the Master of Computer Science (M.C.S.) and M.S. degrees in computer science from Mohammad Ali Jinnah University, Karachi, Pakistan. He is currently pursuing the Ph.D. degree in information technology with Universiti Kuala Lumpur—Malaysian Institute of Information Technology (UniKL MIIT), Kuala Lumpur, Malaysia. He is also a Senior Lecturer with DHA Suffa University, Karachi. He has number of years' experience in industrial and academic field. His research interests include the Internet of Things (IoT), cyber security, and machine learning.

MEGAT F. ZUHAIRI received the B.Eng. degree (Hons.) in electrical and electronics from UNITEN. He has been an Associate Professor with University Kuala Lumpur, Malaysian Institute of Information Technology, since 2004. Prior to that, he was an Engineer with Marconi (M) Bhd., where he worked two years after he completed the B.Eng. degree. He was a Cisco Certified Network Associate (CCNA), between 2007 and 2009. He is currently an Instructor for various courses with Cisco Network Academy. Over the span of ten years, he has published more than 60 journals and conference publications, including two books. His research interests include computer data networking and wireless mobile ad-hoc communications. In recent years, he is more active in blockchain and process mining technology and data acquisition and analysis within the context of Internet of Things. He has served on roughly 24 conference program committees as a reviewer and a technical member.

SYED MUBASHIR ALI received the B.S. degree in computer engineering from NUCES-FAST Karachi, the M.S. degree in information technology from SZABIST Dubai, and the joint Ph.D. degree in computer science from IoBM Karachi and Universiti Kuala Lumpur, Malaysia. He is currently a Professional Engineer, a Computer Scientist, and an Academician with around 12 years of work experience in the fields of IT administration, teaching, and scholarly research. He is also an Assistant Professor and the Head of the Software Engineering Department, Muhammad Ali Jinnah University (MAJU), Karachi. He has published more than 25 research publications in international peer-reviewed journals and conferences. His research interests include multi-criteria decision making, circular economy, human–computer interaction, big data, soft computing, and supply chain management.

ZEESHAN SHAHID received the B.E. degree from the Usman Institute of Technology (UIT) and the M.Sc. and Ph.D. degrees in electrical and electronics engineering from International Islamic University Malaysia. He has published number of research articles in high quality international scientific journals and conference proceedings. He has numerous years' experience in industrial and academic field. His research interests include power engineering specialized in grid-tied inverters, multi-level inverters, DC–DC converters, integration of renewable energy sources (RES) with utility grids, power quality improvement, and electric vehicles.

MUHAMMAD MANSOOR ALAM received the M.S. degree in system engineering, the M.Sc. degree in computer science, the Ph.D. degree in computer engineering, and the Ph.D. degree in electrical and electronics engineering in France, U.K., and Malaysia, respectively, and the Très Honorable degree (Hons.) from Universite de LaRochelle. He was the Associate Dean of CCSIS and the Head of the Department of Mathematics, the Department of Statistics, and the Computer Science Department, IoBM, Pakistan. He is recently associated with Riphah International University, Islamabad. He is currently a Professor in computer science. He has honor to work as an Online Laureate (Facilitator) for MSIS Program run by Colorado State University, USA, and Saudi Electronic University, Saudi Arabia. He also established research collaboration with Universiti Kuala Lumpur (UniKL) and Universiti Malaysia Pahang (UMP). He is also an Adjunct Professor with UniKL and supervising 12 Ph.D. students. He has done postdoctoral research on "Machine learning approaches for efficient pre-diction and decision making" in Malaysia. He is enjoying 20 years of research and teaching experience in Canada, England, France, Malaysia, Saudi Arabia, and Bahrain. He has authored more than 150 research articles which are published in well reputed journals of high impact factor, Springer link book chapters, Scopus indexed journals, and IEEE conferences.

MAZLIHAM MOHD SU'UD received the master's degree in electrical and electronics engineering from the University of Montpellier, in 1993, and the Ph.D. degree in computer engineering from Université de La Rochelle, in 2007. From 2013 to 2020, he was the President/CEO of Universiti Kuala Lumpur, Malaysia. Since 2020, he has been the President/CEO of Multimedia University, Malaysia. He has vast experience in publishing articles in high-quality international scientific journals and conference proceedings. He has numerous years of experience in the industrial and academic fields.