sensors

MDPI

Article

A Trust Management Model for IoT Devices and Services Based on the Multi-Criteria Decision-Making Approach and Deep Long Short-Term Memory Technique

Yara Alghofaili [1] and Murad A. Rassam [1,2,*]

[1] Department of Information Technology, College of Computer, Qassim University, Qassim 52571, Saudi Arabia; 411207305@qu.edu.sa
[2] Faculty of Engineering and Information Technology, Taiz University, Taiz 6803, Yemen
* Correspondence: M.Qasem@qu.edu.sa

Abstract: Recently, Internet of Things (IoT) technology has emerged in many aspects of life, such as transportation, healthcare, and even education. IoT technology incorporates several tasks to achieve the goals for which it was developed through smart services. These services are intelligent activities that allow devices to interact with the physical world to provide suitable services to users anytime and anywhere. However, the remarkable advancement of this technology has increased the number and the mechanisms of attacks. Attackers often take advantage of the IoTs' heterogeneity to cause trust problems and manipulate the behavior to delude devices' reliability and the service provided through it. Consequently, trust is one of the security challenges that threatens IoT smart services. Trust management techniques have been widely used to identify untrusted behavior and isolate untrusted objects over the past few years. However, these techniques still have many limitations like ineffectiveness when dealing with a large amount of data and continuously changing behaviors. Therefore, this paper proposes a model for trust management in IoT devices and services based on the simple multi-attribute rating technique (SMART) and long short-term memory (LSTM) algorithm. The SMART is used for calculating the trust value, while LSTM is used for identifying changes in the behavior based on the trust threshold. The effectiveness of the proposed model is evaluated using accuracy, loss rate, precision, recall, and F-measure on different data samples with different sizes. Comparisons with existing deep learning and machine learning models show superior performance with a different number of iterations. With 100 iterations, the proposed model achieved 99.87% and 99.76% of accuracy and F-measure, respectively.

Keywords: trust management; Internet of Things services; deep long short-term memory; multi-criteria decision-making; simple multi-attribute rating

Citation: Alghofaili, Y.; Rassam, M.A. A Trust Management Model for IoT Devices and Services Based on the Multi-Criteria Decision-Making Approach and Deep Long Short-Term Memory Technique. *Sensors* **2022**, *22*, 634. https://doi.org/10.3390/s22020634

Academic Editors: Christos Xenakis and Thanassis Giannetsos

Received: 29 November 2021
Accepted: 10 January 2022
Published: 14 January 2022

1. Introduction

The rationale behind the Internet of Things (IoT) paradigm was proposed way back in the 1980s with the idea of ubiquitous computing [1]. However, the term IoT received significant attention after the study by [2]. IoT's objective is to incorporate technology into everyday life. Today, the modern IoT environment includes networking and social interactions between physical and cyber components [3]. IoT infrastructure helps run several innovative services (called IoT services) on various platforms, where a large number of heterogeneous devices work together to achieve a common objective [4]. IoT services are used to perform the actual sensing or actuation tasks [5]. In recent years, increasing focus has been given to IoT services in various fields, as seen in Figure 1, which can play an essential role in facilitating humans' daily life [1].

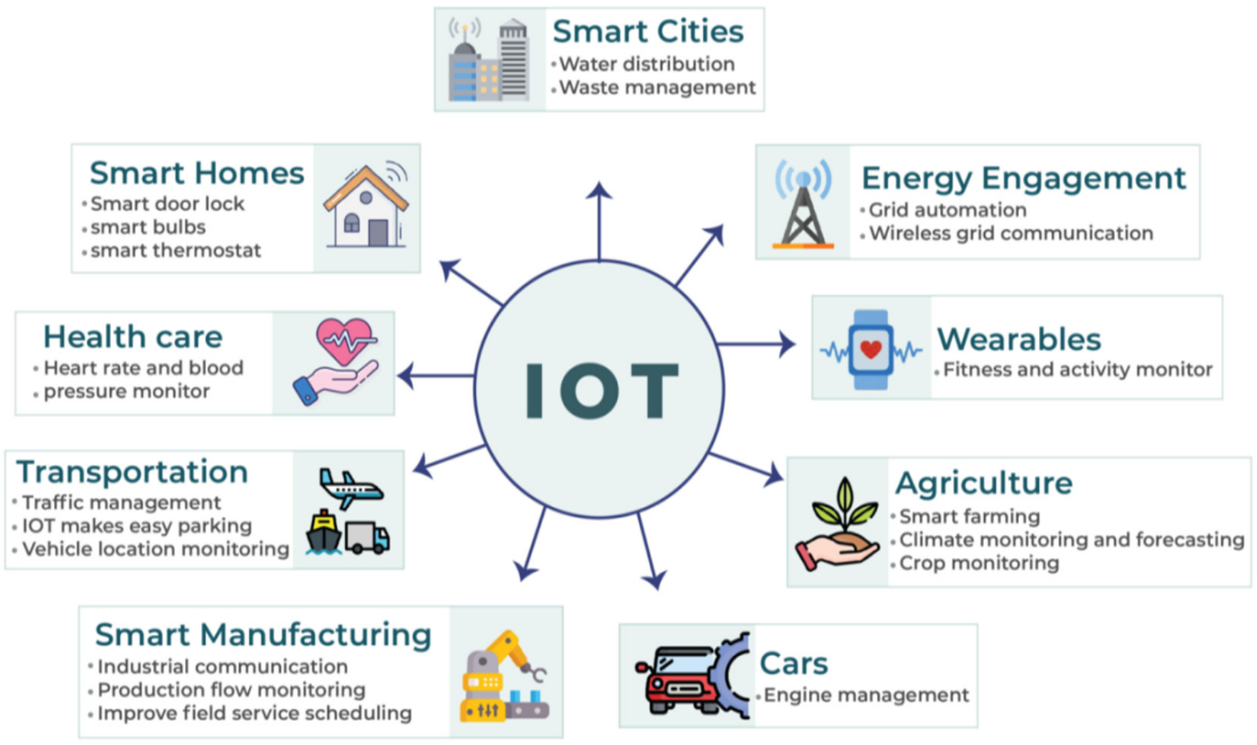


Figure 1. IoT applications and services.

The features that IoT has, such as the diversity of the shared data, dynamicity, and the heterogeneity of devices, bring an entirely new challenge to IoT services and devices. These challenges are mostly solved by considering the security issues in general without evaluating the subjective risks between IoT entities and services [6]. It might result in catastrophic harm and unknown dangers if the information is used for malicious purposes. The principle of trust in IoT can therefore be regarded as a critical feature for establishing trustworthy and reliable service provisioning between different objects [7]. As a result, trust has become one of the major requirements to achieving security.

Trust is an abstract notion, with varying definitions depending on both participants and situations, and informed by measurable and non-measurable variables [7]. This fact indicates that trust is a very complex concept that refers to other factors, such as the ability, strength, reliability, goodness, availability, or other characteristics of an object [8]. Trust management is, therefore, more challenging than security itself, especially in the emerging information technology field, such as IoT [9]. The concept "trust in IoT" refers to the examination of the behavior of devices linked to the same network. The trust connection between two devices influences the future behavior of their interactions. When devices trust each other, they prefer to share services and resources to some degree. Trust management enables the computation and analysis of trust among devices in order to make appropriate decisions for establishing efficient and reliable communication among devices.

Trust management is considered to be a viable solution for IoT trust issues. Such solutions have been used to optimize protection, support decision-making processes, identify untrusted behavior, isolate untrusted objects, and redirect functionality to trusted zones [10]. Various approaches, such as [11–14], have been developed by researchers as solutions to trust issues. However, these solutions are still unable to fully address trust issues and face numerous challenges, such as a lack of effectiveness when dealing with large amounts of data and constantly changing behaviors, high energy utilization, difficulty in quantifying uncertainty for untrusted behaviors, choosing the optimal trust model components, and dealing with IoT's dynamic nature and heterogeneity.

Aimed at the former issues, this paper proposes a trust management model for IoT devices and services that takes leverage from multi-criteria decision-making and deep learning techniques. This model can identify suspicious activities and take appropriate actions, such as isolating untrusted entities and redirecting IoT functionality to trustworthy zones. The main contributions of this paper include: (1) creating a new dataset by extracting more features from real packet captures and patches that already exist in the literature; (2) calculating the trust value using the simple multi-attribute rating technique (SMART), which determines the value of the trust depending on the node information only to reduce the risk of threats that result from wrong recommendations. Additionally, it reduces the energy computation that makes the algorithm lightweight; and (3) Developing an intelligent solution based on the long short-term memory (LSTM) technique that counters the continuous change in behaviors as well as being compatible with big data to ensure stockholders benefit from these integral and available services of IoTs.

The rest of this paper is organized as follows: a background on trust management is presented in Section 2. The related works are investigated and discussed in Section 3. The proposed model is described in Section 4. The experimental investigation, result analysis, and evaluation of the proposed model are reported in Sections 5–7 reports the comparison results with existing models and Section 8 concludes the paper and suggests some future research directions.

2. Trust Management Principles and Terminologies

In this section, the main components of building a trust management model are introduced. These components are referred to as computational trust modules and are used to quantify and evaluate entity attributes, such as integrity, reliability, honesty, and others, to estimate the value of trust [6]. Figure 2 shows the five components, and the subsequent sections describes these components.

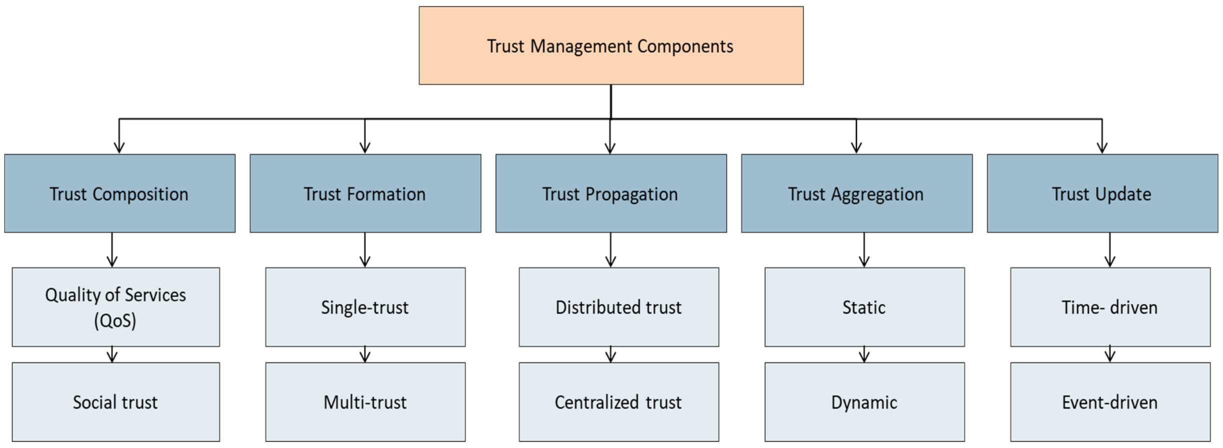


Figure 2. Trust management model components.

2.1. Trust Composition

This refers to the components that are considered in trust computation and it involves two major modules namely: quality of service (QoS) trust and social trust [8].

(1) QoS refers to the expectation of an IoT entity to provide superior quality in its functionalities. QoS trust utilizes some trust properties, such as competence, reliability, task completion capability, and cooperativeness, to measure the value of trust [8].

(2) Social trust refers to a social relationship among IoT entity owners. Social relationship trust is used to assess the IoT entity to evaluate whether it is trustworthy or not. Besides, social trust utilizes trust properties, such as honesty, centrality, intimacy, privacy, and connectivity, to measure trust values [8].

2.2. Trust Formation

This refers to whether trust computation is based on either one trust attribute (single-trust) or the use of multiple attributes (multi-trust). Besides, these components are chiefly concerned with what weights are put on QoS and social trust attributes from trust [8].

2.3. Trust Propagation

Trust propagation refers to the way of propagating trust information to other entities. Under this form of propagation [6], two main schemes can be identified:

1. Distributed trust refers to IoT entities autonomously propagating trust and observations to other IoT entities they interact with or encounter without the necessity for a centralized entity [6].
2. Centralized trust requires the presence of centralized entities. It can exist as either a virtual trust service or a physical cloud that is implemented by IoT devices [6].

2.4. Trust Aggregation

This refers to the most appropriate method of aggregating trust information, which is then evaluated by the entity itself (direct evaluation) or by other entities (indirect evaluation) [8]. This component aggregates information using weights, which might be static or dynamic. The static is calculated in accordance with the entity attributes. The original trust on both communication parties is built on both sides' trust attributes. To make proper dynamic trust decisions, trust management must rely on context information when assigning weights to each property [15]. In the literature, there are various models of trust aggregation including the belief theory, fuzzy logic, Bayesian inference, weighted sum, and regression analysis [8].

2.5. Trust Update

This component decides when to update the values of trust. The updating of the trust information occurs periodically (time-driven) by applying a trust aggregation or after a transaction or event affects the QoS (event-driven) [8].

3. Related Works

Adopting trust management solutions is one of the promising trends that addresses the challenges raised by suspicious IoT devices and services. For instance, a study in [16] suggested a protocol for trust management consisting of three variables: cooperativeness, honesty, and community interests. Using this protocol, it was possible to create new nodes with the intent of establishing trust relationships with other nodes and endure in unsafe environments. Another study in [17] integrated two models (subject and object) to make a reliable system in regard to the objects' performance. In the first model, each node calculated its friends based on its experiences and the friends' thoughts in common with the potential providers to adapt behavior dynamically. In the other model, each node's data was assigned and stored using a Distributed Hash Table structure; therefore, any node can utilize similar data.

A further study by [18] developed a trust propagation model for IoT services. The model depended on dispersed collaborative filtering to arrive at the feedback by utilizing social contact, similarity ratings of friendship, and interest relationships while using community as the filter. Both studies by [3,19] introduced fuzzy logic-based trust assessment methods. The first study used the Bio-inspired Energy Efficient-Cluster (BEE-C) protocol and fuzzy logic to compute the trust of the nodes. The value of the trust was compared to the threshold value. Trust values above the threshold were considered to be trusted nodes. Likewise, the trust value below the threshold value was defined as a non-trusted node and was eliminated. In contrast, the other study used fuzzy to solve the network traffic that influences energy dissipation through the data transmitted by sensor nodes. The scheme implemented decision-making to authenticate the sensor nodes of the network to perform a trusted aggregator.

Moreover, the study by [14] designed a simple trust management model founded on entropy and the Bayesian principle. The Bayesian principle is used to compute the value of the trust for direct nodes and periodically updated. Similarly, entropy theory distributes weights to various values of trust that can enhance the issues caused by subjectively distributing weights and also improve a model's adaptability. Similarly, a study by [13] developed a distributed trust management model in IoT. The model's goal was to detect malicious node activity and avoid potential on-off attacks on a multi-service IoT. Three phases of the model are contained, which are neighbor discovery, service request, and trust computation.

Furthermore, a study by [12] suggested an IoT trust and reputation-based recommendation method using a probabilistic neural network (PNN). It was performed on IoT edge devices to distinguish between untrustworthy and trustworthy nodes. The model solved the initial value trust issue in IoT environments by forecasting ratings based on the attributes for new devices and learning over time. Another study by [20] suggested a Central Trust management framework for the Internet of Things (CTM-IoT) to provide trustworthy information exchange across IoT devices. The concept included a super-node that served as a centralized trust manager. The trust information of all master nodes and cluster nodes was stored in the central repository by the super-node. The super-node was also in charge of monitoring different activities, such as network traffic and trust management, across all IoT devices. Additionally, the super-node contained a repository in which all master node trust values and addresses were kept. The repository acted as a routing table, recording trustworthy information as well as the network structure, and controlled all devices in the CTM-IoT framework, determining which devices must join which cluster.

Besides, [7] presented a computational trust model for IoT services based on machine learning techniques. This model used two techniques: (i) k-means for clustering and labeling tasks, such as identifying the number of clusters and initial centroid positions; and (ii) support vector machine (SVM) for classification tasks, such as identifying the boundaries of trustworthy and untrustworthy interaction. Similarly, a study in [21] suggested smart algorithms to manage IoT trust. The first algorithm suggested a new clustering method by calculating memory boundary trust value limits for each cluster while the second algorithm established conditions under which a cluster node in IoT trust management can be changed to a specified new master node. The third algorithm is used to address the bad-mouthed attacks. The fourth algorithm proposed methods by which master nodes track trust values for cluster nodes and attempt to shift some cluster nodes away.

A further study [22] proposed a fuzzy logic-based protocol for detecting on-off attacks, contradicting behavior attacks, and other bad nodes. This protocol allowed nodes to transfer securely from one cluster to another. Furthermore, for secure message encryption, it employed a messaging system similar to serial transmission. Additionally, the protocol utilized fuzzy logic to identify bad nodes and limit their untrusted role of making erroneous recommendations regarding nodes in the network.

A study [11] developed a model that utilizes various parameters, such as the device ownership trust, device security, and level of security in a device, to determine the trust level based on the fuzzy logic model. The fuzzy logic model was used to assess the degree of trust with using the threshold selected by users. IoT service users can also play an active role in the selection process of their trusted nodes tasked with collecting their data when the trust level is higher than the threshold. Moreover, the research by [23] suggested a trust assessment model using multi-layer perceptron (MLP). This model allowed the types of trust-related attacks carried out by malicious nodes to be detected and separated from the network to achieve a secure environment.

Another study by [24] developed a smart device selective recommendation method that utilizes a dynamic black-and-white list. The aim of this method is to eliminate the problem faced when selecting participants in order to improve the quality of services offered by edge computing systems utilizing IoT in a smart city. Game theory was introduced to qualitatively analyze the stability and validity of the proposed trust management system.

In addition, Lyapunov theory was used to verify results obtained from game theory. A recent study by [25] proposed a trust framework using a neural network. This framework considered many perspectives (e.g., owner, device, and service) and each perspective considered certain attributes (e.g., social and locality for the owner, reputation of the device, and reliability of the service).

Similarly, a study in [26] developed an approach for trust management in social IoT devices. The approach contained three main stages: (1) trust composition stage—at this stage, various attributes are chosen as per the attack context. For the trust computation process, the trustee node was selected from the set of nodes based on trust attributes; (2) aggregation stage, through this stage, the trust score was calculated based on the artificial neural network (ANN) algorithm; and finally (3) the update stage, the time-driven model was used to update the trust score periodically. A study conducted by [27] recommended a dynamic trust management mechanism in wireless sensor networks (WSNs). Firstly, the node's direct trust value is determined by evaluating its performance from interaction with regional information. After, the comprehensive trust value is calculated using the energy evaluation and trust recommendation value of other nodes with a high trust level. Finally, the node management and reliability of nodes are periodically updated.

Recently, a study by [28] suggested an information entropy-based trust evaluation approach to solve the issue of trust in the power distribution of communication terminals in Internet of Things. First, the model estimated the direct trust value based on the reputation of an exponential distribution, and then, the forgetting factor and sliding window updated the direct trust value. Uncertainties in the direct trust value were assessed, and the indirect trust value was added to compensate for inaccuracies that arise from direct trust judgment. In addition, the indirect and direct trust value were assessed completely to enhance the judgment accuracy.

Table 1 summarizes the related current techniques used for trust management in IoT devices and services based on the design components for each approach.

Table 1. Summary of existing studies.

Approach	Components of Design Trust Management								
	Trust Composition	Trust Formation		Trust Propagation		Trust Aggregation		Trust Update	
		Single-Trust	Multi-Trust	Centralized	Distributed	Static	Dynamic	Time-Driven	Event-Driven
[16]	√	x	√	x	√	x	x	√	x
[17]	√	x	√	x	√	√	x	x	x
[18]	√	x	√	x	√	x	√	√	x
[19]	√	x	√	x	√	√	x	x	x
[14]	x	x	√	x	√	x	√	√	x
[13]	x	x	√	x	√	√	x	x	x
[3]	x	x	√	x	√	√	x	x	x
[12]	√	x	√	x	√	x	√	√	x
[20]	x	√	x	√	x	x	x	x	√
[7]	√	x	√	x	√	x	√	x	x
[21]	x	x	x	√	x	x	√	√	x
[22]	x	x	x	√	x	x	√	x	√
[11]	√	x	√	x	√	√	x	x	x

Table 1. *Cont.*

Approach	Components of Design Trust Management								
	Trust Composition	Trust Formation		Trust Propagation		Trust Aggregation		Trust Update	
		Single-Trust	Multi-Trust	Centralized	Distributed	Static	Dynamic	Time-Driven	Event-Driven
[23]	√	x	√	x	√	x	√	√	x
[24]	x	√	x	x	√	x	√	x	√
[25]	√	x	√	x	√	x	√	x	x
[26]	√	x	√	x	√	x	√	√	x
[27]	x	x	√	x	√	x	x	√	x
[28]	x	x	x	x	√	√	x	√	x

To conclude, managing trust is a critical problem that is seen as a significant challenge for IoT devices and services. Several solutions have been proposed in the literature as discussed earlier in this section. However, some serious research gaps are still unsolved, which are summarized in the following sections.

3.1. Diversity in Components of Trust Models

Studies have provided clear evidence that trust is a very complex concept and has various meanings, since they have addressed the trust issues depending on the author's view. As can be seen in Table 1, many studies have used a variety of components while developing trust models. Therefore, the components of design trust management differ depending on the study.

In the trust composition component, the existing studies calculated this component using either quality of service (QoS) (which is done by utilizing some trust properties, such as competence, reliability, task completion capability, and cooperativeness, to measure the value of the trust [8]) or social trust (this is done by utilizing trust properties, such as honesty, centrality, intimacy, privacy, and connectivity, to measure trust values [8]). Thus, selection of the optimal components can be a challenge for managing trust. Most studies have evaluated trust management models according to the risk and logic only and ignored the composition components, which play a significant role in assessing the relationships between entities or between behaviors [29].

In both trust formation and propagation, most studies have relied on more than one attribute to measure trust. Multiple attributes make the model more accurate because the evaluation will depend on more than one feature. In addition, many studies in the literature relied on the use of a distributed model that gives trust values to the node itself without the need for an intermediate central node. Therefore, the use of distributed models is better, because the process of assigning the trust value to each node is faster, and in the case of an unauthorized attack, it reduces the risk of infection for the rest of the nodes in the system.

In terms of the aggregation component, most existing studies used both direct and indirect trust to obtain the information. However, indirect trust may cause many problems, such as incorrect recommendations and high computational capacity and time need to assign trust values, in contrast to direct trust. Besides, most studies have focused on static aggregation components, which may not be effective with the dynamic nature of IoT. Although the studies have worked on machine learning algorithms that may improve aggregation processes, they focused on the dynamics of the aggregation process itself and assigning weights manually, as is the case in fuzzy logic, which depends entirely on human experience and knowledge. Therefore, employing dynamic aggregation with dynamic assigning of weights helps to enhance data efficiency and accuracy [30].

In terms of the trust update component, updating the trust is important as it identifies changes to the node after a specific event or time. Most of the studies either ignored this component or depended on the time-driven approach while designing their models. Few studies adopted the event-driven approach. However, updating the trust based on the event is essential for some reasons, such that after any event, the trust values of the node may increase or decrease according to its behavior. Therefore, it is illogical to rely on a specific time to measure the trust of nodes that may be infected from the previous event.

3.2. Shortcomings of Techniques Used

The attackers mostly use untrusted entities to manipulate their behaviors and act as trustworthy entities; identifying these misbehaviors is essential. Existing studies have focused on solving a specific type of malicious behaviors, but an advanced attacker may choose a sophisticated strategy to act maliciously. Consequently, most existing research and development efforts in the domain of trust management are centered around applying statistical models or machine learning techniques, such as in [3,7,14,22,23,27]. These techniques have several drawbacks, including ineffectiveness when dealing with big data and continuously changing behaviors, high memory utilization, and difficulty in quantifying uncertainty for untrusted behaviors. Consequently, deep learning approaches may become an excellent alternative for overcoming the mentioned constraints of machine learning and statistical techniques. Deep learning has found widespread use in computer vision, speech recognition, robotics, and misbehavior detection, among a variety of other fields [31]. Deep learning offers several benefits over machine learning and statistical techniques: (1) deep learning can match complicated nonlinear connections between variables due to the usage of numerous hidden layers inside a neural network topology, (2) it is also especially well adapted to coping with "big data" issues, and (3) it is able to teach IoT devices complex behavioral patterns more successfully than machine learning and statistical techniques [31].

4. Proposed Model

The proposed model is divided into four main stages: data collection, data preparation, trust prediction, and evaluation. Figure 3 depicts the architecture of the proposed model.

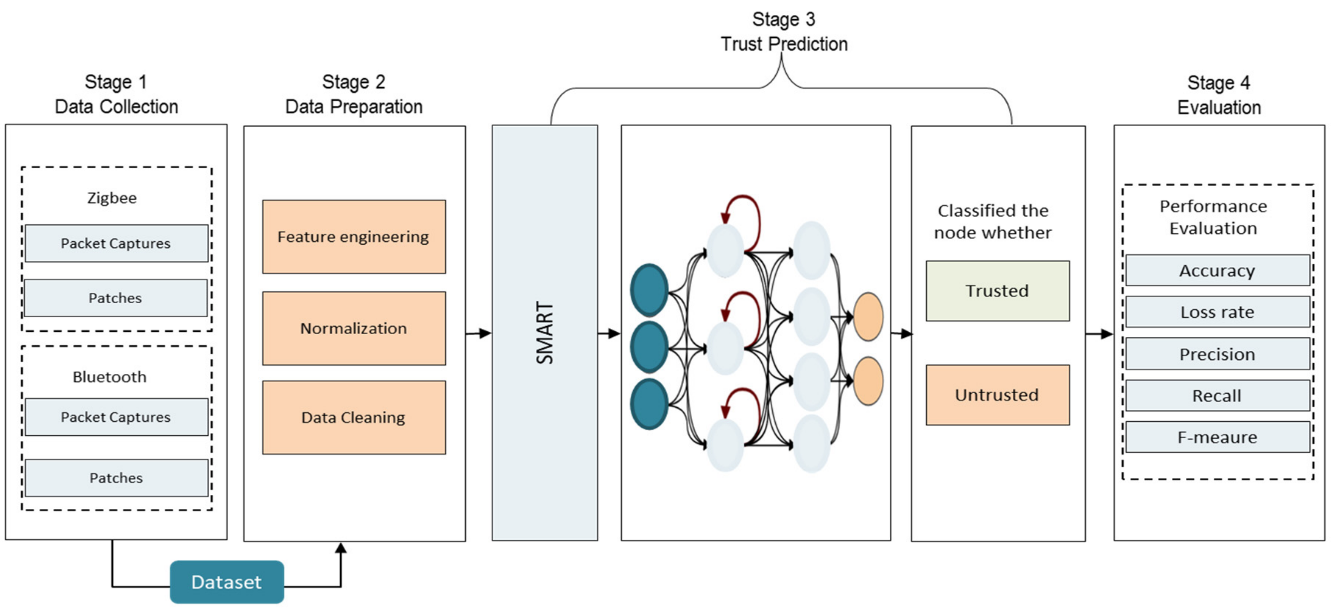


Figure 3. Proposed Model.

4.1. Data Collection Stage

This stage collects the data to test the model in the next stages. This study uses packet captures and patches proposed by [32]. The data is collected from IoT devices' activity utilized to monitor smart homes for 10 days. IP (including Ethernet, Wi-Fi, and PPP), Bluetooth, Z-Wave, RF869, and ZigBee protocols are installed. Tables 2 and 3 show the details of the devices and the number of captures and batches. For packet captures, it contains the information about the source and destination addresses, timestamp, data and packet length, the destination PAN id, and data. In the patches, it contains the information about the source and destination addresses, timestamp for the start and end, duration, packets number, and size of the packet.

Table 2. Device deployment places [32].

Device Type	Protocol	Placement
Motion sensor	Zigbee	Living room
Motion sensor	Zigbee	Kitchen
Motion sensor	Zigbee	Bathroom
Motion sensor	Zigbee	Bedroom
Door sensor	Zigbee	Entrance door
Door sensor	Zigbee	Dishwasher
Weight scale	Bluetooth	Nearby the gateway
Blood pressure meter	Bluetooth	Nearby the gateway
Gateway	Bluetooth	Office
Gateway	Zigbee	Office

Table 3. Number of packets and patches for each protocol [32].

Protocol	Packet Captures	Patches
Zigbee	73,876	27,385
Bluetooth	541,544	22,202

4.2. Data Preparation Stage

During this stage, many sub-stages are used for data preparation, such as feature engineering, normalization, and cleaning.

4.2.1. Feature Engineering

The primary goal of feature engineering is to create or extract features from existing data [33]. Therefore, at this sub-stage, some of the existing features are used to create additional features (e.g., packet loss, delay, and throughput). The following definitions and equations are according to [34].

Packet Loss—The failure of packets to reach their destination is referred to as *packet loss*. The value of *packet loss* can be calculated using Equation (1):

$$Packet\ Loss = \frac{Packet\ sent - Packet\ recived}{Packet\ sent} \times 100 \tag{1}$$

Delay—The latency caused by *transmission* from one point to another, which becomes the goal, is known as a *delay*. Equation (2) is used to calculate the delay:

$$Delay = propagation\ delay + transmission\ delay + queuing\ delay + processing\ delay \tag{2}$$

Propagation Delay—The amount of time it takes for a bit to travel from the source to its destination. The *propagation delay* is computed by dividing the *distance* by the *propagation speed*, as shown in Equation (3):

$$Propagation\ delay = \frac{distance}{Propagation\ Speed} \tag{3}$$

where the *distance* is the average *packet* size * 1000 and the *propagation speed* is the constant value (=3 $\times$ 10^8 m/s).

Transmission Delay—The amount of time it takes to send a *packet* from the source to the *transmission medium*, as shown in Equation (4):

$$Transmission\ delay = \frac{length\ of\ packets}{bandwidth} \tag{4}$$

where *bandwidth* represents the maximum number of *packets*.

Queuing Delay—This delay is caused by the time needed for routers to handle packet transmission queues across the network.

Processing Delay—The time it takes a network device to see the route, update the header, and switch tasks is referred to as processing delay.

Throughput refers to the actual *bandwidth* that was measured at a specific time and under specific network conditions to transfer files of a specific size. The total speed of data sent to all terminals in a network is known as throughput, which can be calculated using Equation (5):

$$Throughput = \frac{\sum Packet\ sent(bits)}{Time\ of\ data\ deilvary\ (s)} \times 100 \tag{5}$$

4.2.2. Normalization

In this process, the features are scaled to values ranging from 0 to 1 to produce an accurate result. This step is necessary to transform the numeric column values in the dataset; therefore, it may be used on a common scale without distorting the variation in value ranges or losing data [35]. The normalization is performed using Equation (6):

$$z_i = \frac{x_i - \min(x)}{\max(x) - \min(x)} \tag{6}$$

where x_i is the dataset's ith value, min(x) is the dataset's minimum value, and max(x) is the dataset's maximum value.

4.2.3. Data Cleaning

This sub-stage aims to clean the data by assuring the validity of dataset samples, such as removing the null and negative values in records.

4.3. Trust Prediction Stage

The trust prediction stage is divided into two sub-stages: the calculation of the trust value and misbehaving detection. In the trust value calculation sub-stage, the simple multi-attribute rating technique (SMART) is used, which determines the value of the trust based on the node information extracted in the previous stage (data preparation). In the misbehaving detection sub-stage, the long short-term memory (LSTM) technique is used for classification/prediction tasks, which is known as an excellent technique for identifying changes in behavior. During this sub-stage, the learned model classifies new unknown data (included in the test set) that the model has never seen before to assess the learned model's capabilities (initially, the detective ability of the model is evaluated and, if it is acceptable, then the learned model can be used for detection). The following subsections detail those two sub-stages.

4.3.1. Trust Value Calculation

In this sub-stage, the data is identified as trusted or untrusted using the SMART technique. The SMART technique is a method used for solving multi-criteria decision-making (MCDM) issues. It is founded on the concept that each alternative is composed of a number of criteria with values, and each criterion has a weight that indicates its relevance in comparison to other criteria [36,37]. Figure 4 shows how the SMART calculates the trust value.

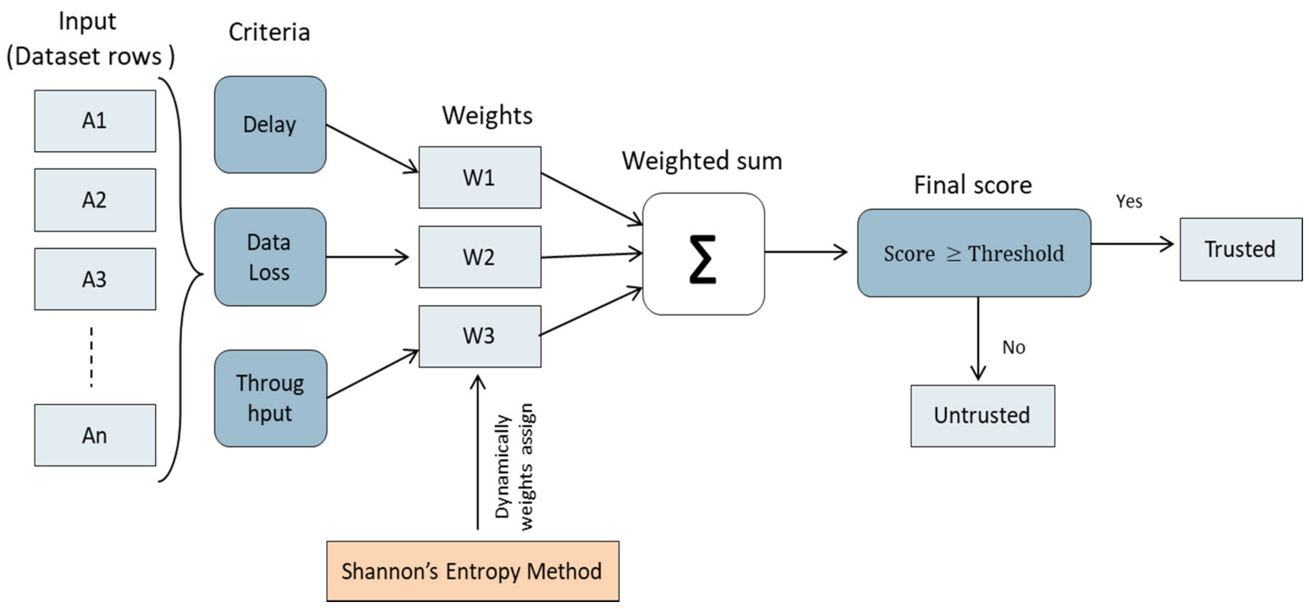


Figure 4. Trust value calculation using the SMART technique.

Step 1: Decision context and structuring includes defining the alternatives and determining the number of criteria that will be utilized.

Step 2: Analysis, which includes the following:

1. Determining the criteria weights for each criterion using the 1 to 100 scale for the criterion using Shannon's entropy method, which is a well-known approach for determining weights for an MADM issue (e.g., static weight assign) [38]. Shannon's entropy method is designed as an objective method of allocating weights based on the decision matrix without impacting the decision maker's preference [39,40].

In this study, a combination of the SMART and Shannon's entropy methods is used to calculate weights dynamically based on the specified criteria. The Shannon's entropy (E_j) can be calculated using Equations (7)–(9). Suppose k_j (j = 1, 2, 3...) includes various alternatives and k_i (i = 1, 2, 3...) represents the criteria inside these alternatives. The ith criteria value in the jth alternative is then indicated by k_{ij}, and the weight evaluation procedure is created on this basis. Because the dimensions of the various alternatives throughout the evaluation are not similar, these factors should be standardized using Equation (7):

$$Rij = \frac{k_{ij}}{\sum_{i=1}^{m}\sum_{i=1}^{n} k_{ij}} \tag{7}$$

where R_{ij} denotes the specific gravity per k_{ij} and m denotes the number of criteria. Then, the entropy for each factor alternative E_j is calculated using Equation (8):

$$E_j = \left[\frac{-1}{\ln(m)}\right] \sum_{i=1}^{m} \left[R_{ij} \ln\left(R_{ij}\right)\right] \tag{8}$$

where m is the number of standardized assessment possibilities in the matrix and ij is the number of criteria.

$$D_j = 1 - E_j \tag{9}$$

where D_j is the diversity criterion.

2. Normalizing each criterion by dividing the number of weighted criteria by the number of weights, using Equation (10):

$$\text{Wj} = \frac{D_j}{\sum_{j=1}^{k} D_j} \tag{10}$$

where D_j is the weight value of the criteria, $\sum_{j=0}^{k} D_j$ is the total weight of all criteria, and j is the number of possibilities from 1 to k.

3. Providing a value for each criterion parameter for each alternative.

Step 3: Decision, which involves the following:

1. Determining the utility value to transform the value of each criterion‘s criteria into the value of the raw data criteria. Equation (11) is used to calculate the utility value:

$$u_i(a_i) = \frac{c_{out} - c_{min}}{c_{max} - c_{min}} \tag{11}$$

where $u_i(a_i)$ denotes the utility value of the criterion to i for the criterion to j, c_{max} is the greatest criterion value, c_{min} is the lowest criterion value, and c_{out} is the criteria value of i. The significance of these values is shown in Equation (12):

$$c_{out}\ i = u_i(a_i),\ \ 1 = 0;\ 2 = 0.5;\ 3 = 1 \tag{12}$$

Equation (11) is used to determine the value of the utility to convert the value of the criterion to i one of the criteria to i. The computation then yields the following results:

- If the criteria value (c_{out}) = 3, then $u_i(a_i) = \frac{3-1}{3-1}$ = 1;
- If the criteria value (c_{out}) = 2, then $u_i(a_i) = \frac{2-1}{3-1}$ = 0.5;
- If the criteria value (c_{out}) = 1, then $u_i(a_i) = \frac{1-1}{3-1}$ = 0.

2. Determining the final value of each criterion by shifting the values obtained from the normalized value of the raw data criteria with weight normalized value criteria, using Equation (13):

$$u(a_i) = \sum_{j=1}^{n} w_j\, u_i\ \text{, where } i = 1, 2, \ldots\ldots, n \tag{13}$$

3. Calculating the dynamic threshold (DT) using the mean absolute error (MAE). MAE is a metric that measures how close predictions are to actual results. MAE is employed because it gives a straightforward means of determining the degree of significance of errors [41]. It is commonly used in the security field to quantify errors depending on the problem. In particular, it is used in trust management to determine the threshold or ground value, as defined in [42,43]. In this paper, the MAE is used for DT, as defined in Equation (14):

$$\text{Dynamic Threshold (DT)} = \frac{\sum_{i=1}^{n} \left| u(a_i) - \overline{u(a_i)} \right|}{n} \tag{14}$$

where $u(a_i)$ is the value of the trust, $\overline{u(a_i)}$ is the predicted trust value, and n is the total number of samples.

4. Comparing the value of trust with the DT that was obtained using Equation (15), if the value of trust is greater or equal to the DT value, this means the device is trusted; otherwise, it is untrusted:

$$Trust\ Score = \begin{cases} u(a_i) < \mathrm{DT}, & Untrused \\ u(a_i) \geq \mathrm{DT}, & Trusted \end{cases} \tag{15}$$

4.3.2. Misbehaving Detection

In this sub-stage, the long short-term memory (LSTM), a deep learning technique that has recently sparked the interest of the research community, is used. LSTM has produced excellent results when applied to complicated issues, such as the translation of languages, text generation, and automatic captioning of images, among other applications [44]. This technique has been widely used in recent years to overcome security issues, such as in [45–47]. For this reason, this study uses LSTM to detect malicious behaviors that may indicate trust violation issues.

4.4. Evaluation Stage

This stage includes a detailed experimental assessment of the proposed model. There are several metrics for various algorithms, and these metrics have been devised to evaluate the effectiveness and efficiency of the models. The false positive (FP), false negative (FN), true positive (TP), and true negative (TN), as well as their relationships, are parameters often used by trust management researchers to evaluate the performance of various techniques [48]. The definitions of these parameters are as follows:

True Positive (TP) indicates the part of the untrustworthy entities properly identified as untrusted.

True Negative (TN) indicates the part of the trustworthy entities properly identified as trusted.

False Positive (FP) represents the part of the trustworthy entities wrongly identified as untrustworthy.

False Negative (FN) represents the part of the untrustworthy entities wrongly identified as trustworthy.

This study employs five different metrics based on these parameters, which include accuracy, loss rate, precision, recall, and F-measure, to measure how well the model performs.

Accuracy is defined as the degree of agreement between the actual measurement and the absolute measurement. Accuracy is one of the most widely used metrics of classification performance, and it is described as the ratio of properly categorized samples to the total number of samples [49], and is calculated as Equation (16):

$$\mathrm{Accuracy} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}} \tag{16}$$

Loss Rate is a function that quantifies the difference between the actual and predicted output during training to accelerate the learning process. It is also used to evaluate model performance and minimize error [45]. Equation (17) is used to compute the loss rate:

$$\mathrm{Loss} = -\,\mathrm{Y} \times \mathrm{Log}(\mathrm{Y}_{\mathrm{Pred}}) - (1 - \mathrm{Y}) \times\ \mathrm{Log}(1 - \mathrm{Y}_{\mathrm{Pred}}) \tag{17}$$

Precision refers to the classification model's ability to retrieve just data points from a certain class. It is defined as the number of correct samples recovered divided by the total number of samples retrieved [50], and is shown in Equation (18):

$$\mathrm{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{18}$$

Recall is the capacity of a classification model to recognize all data points in a relevant class. It calculates the number of accurate samples retrieved divided by the total number of correct samples [50], and is defined as in Equation (19):

$$Recall = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{19}$$

F-Measure is another measure derived from *precision* and *recall* that represents the behavior of both measures [51]. It is computed as shown in Equation (20):

$$F - \ Measure = 2 \times \frac{precision \times recall}{precision + recall} \tag{20}$$

The implementation of the methodology of this study relies on several trust components, as shown in Table 4. For trust composition and formation, it is done by using multiple features to measure the QoS (*packet loss*, *delay*, and *throughput*). For trust propagation, the model uses decentralized methods to propagate the trust to reduce the loss in case of an attack infection. For the trust aggregation, the model uses both dynamic calculations of processes and weights using SMART and entropy. Finally, for the trust update, the model takes advantage of the benefit of the LSTM at detecting the changes after any event; therefore, it is used for updating purposes.

Table 4. Trust components used in this study.

Trust Component	How It Is Applied in the Implementation Part?	Stage No.
Trust Composition	By measuring the QoS for the obtained features	2 and 3
Trust Formation	Using multi multi-features to calculate the value of the trust	3
Trust Propagation	Calculate the value of trust that means the propagation is decentralized	All model
Trust Aggregation	Using both SMART and entropy to dynamically aggregate the value of features and dynamically calculated the weight of features	3
Trust Update	By applying LSTM cells to detect the changes after an event happened	3 and 4

5. Experimental Investigation

The setup of the model and the dataset description is presented in the following subsections.

5.1. Model Setup

This experiment was carried out on Google CoLab with the help of Python library packages, such as Pandas, Numpy, Scikitlearn, Matplotlib, and Keras, to calculate the trust value and perform the preprocessing task. The misbehaving detection model was created with LSTM cells, drop out, and dense output layers. Table 5 describes the layers and the values of the parameters used. The model was run using 50 and 100 epochs and a batch size of 72. Furthermore, the model employed the Rectified Linear Unit (ReLu) and sigmoid activation functions and Adam optimizer.

Table 5. Model setup settings.

Parameters	Value
Language	Python
Libraries	Pandas, Numpy, Scikitlearn, Matplotlib, and Keras
Train set	70%
Test set	30%
Input Layer	1
LSTM Cells	2 cells
Activation Functions	Rectified Linear Unit (ReLu), and sigmoid
Dense Layer	1
Dropout	0.20
Optimizer	Adam
Number of Epochs	50 and 100
Batch size	72

5.2. Dataset Description

The dataset was split into training and testing sets with a ratio of 70:30, respectively. To avoid over fitting and under fitting, the data was randomly divided several times until it was verified that the testing set represented behaviors that were unseen before.

6. Results and Analysis

This section presents the dataset collection and visualization. Additionally, this section discusses the trust prediction stage experimental results and results analysis. Through this sub-section, the value of trust and misbehavior detection are discussed.

6.1. Dataset Collection and Visualization

As a result of the feature engineering process described in Section 4, three features were obtained, including packet loss, delay, and throughput. Table 6 describes the ranges of each feature, which vary from good, medium, to poor based on the study by [34]. For packet loss, this feature was calculated to detect any changes that may affect the availability of the services and guarantee their reliability. Figure 5 shows the density of loss values for the overall dataset. The delay feature was calculated to test the performance of the network (when the value of delay is high, it will decrease network performance). Figure 6 shows that the delay has varying values, but most of these density values are between 140 and 170. The throughput feature was calculated as forecasted to match the demands of the current network's services. Figure 7 shows the density of the throughput values in the dataset. These features were used as input for the next stage (the trust value prediction stage).

Table 6. Ranges of the selected features.

Range	Feature Name and Its Ranges		
	Packet Loss	Delay	Throughput
Good	Less than 3%	0–150 ms	100%
Medium	More than 15%	151–400 ms	75–50%
Poor	More than 25%	More than 400 ms	Less than 25%

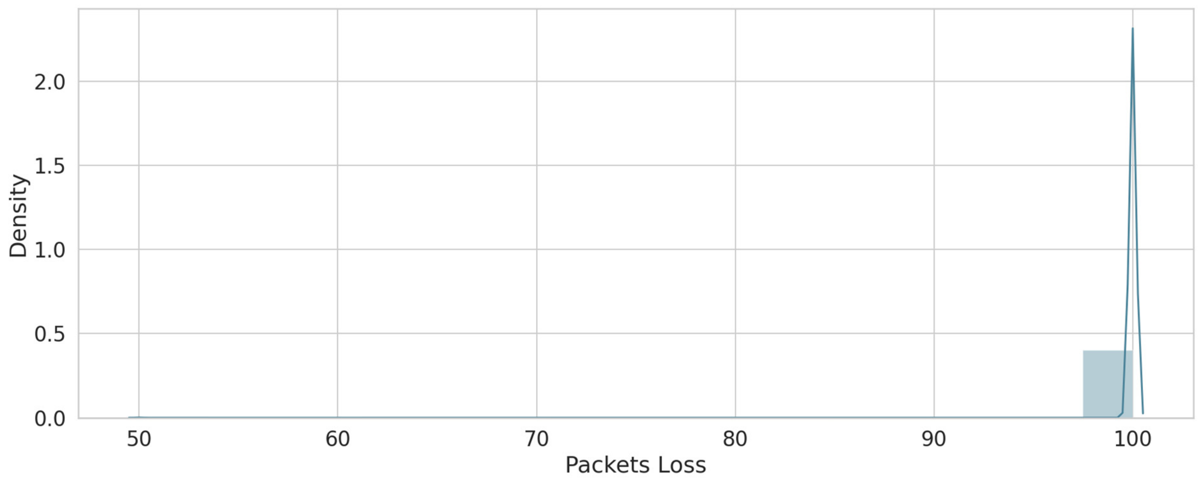


Figure 5. Packets loss sample.

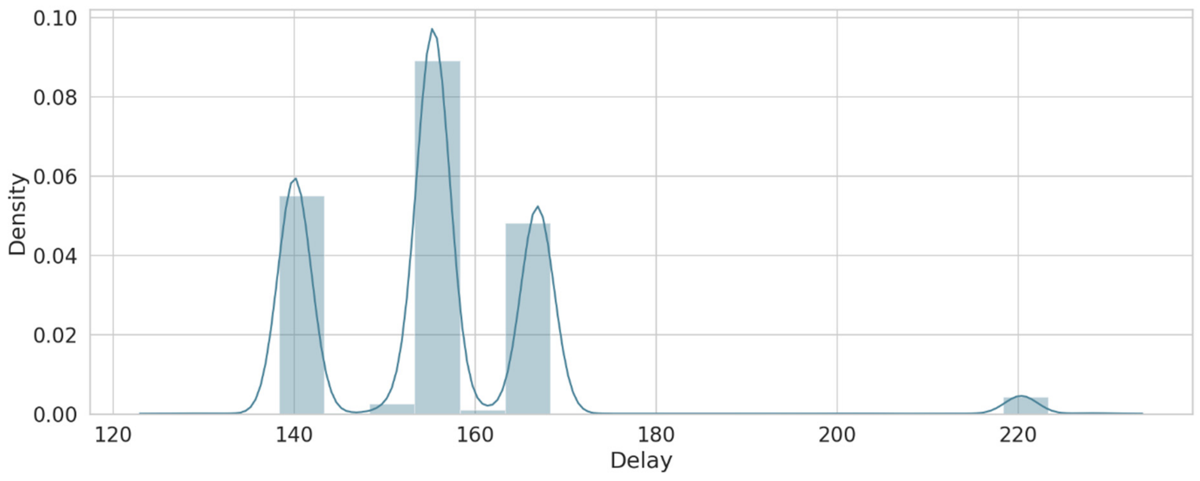


Figure 6. Delay sample.

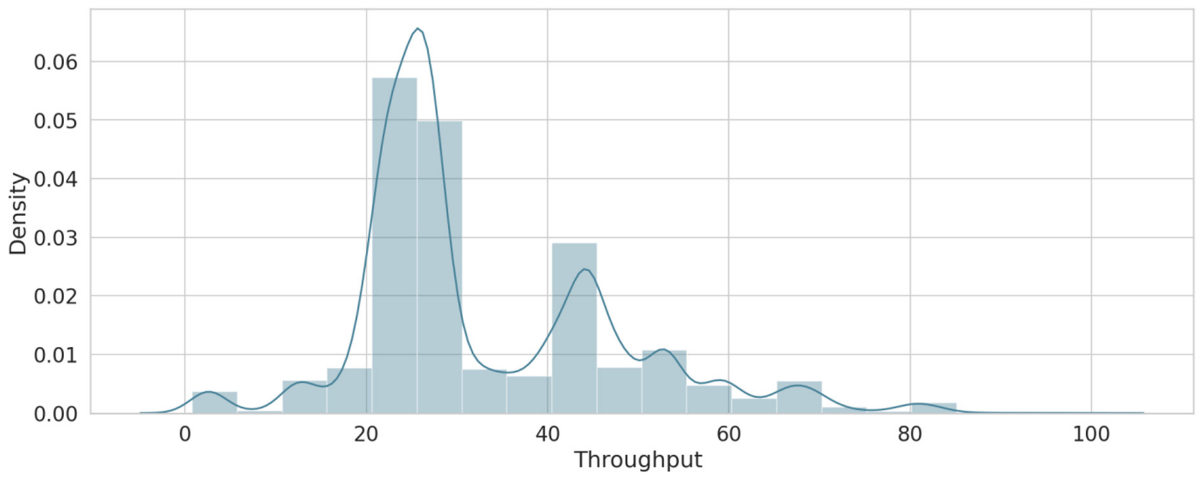


Figure 7. Throughput sample.

6.2. Trust Prediction Results

In this sub-section, the results of the trust value calculation and misbehaving detection are presented.

Trust Value Calculation Results

Step 1: Decision context and structuring: for the dataset used in this study, IoT devices represent the alternatives. Additionally, this study determined whether the device is trusted or not by investigating the three criteria (packet loss, delay, and throughput), as shown in Table 7. Each criterion has a range that indicates how well this device does. These ranges vary from poor, medium, to good, which are denoted as 1, 2, and 3, as shown in Table 8. The range helps to rearrange the values of criteria in the dataset. As seen in Table 9, in C1, which represents the packet loss, if the value in the data is less than 3%, it means the data is good and will be represented by a value of 3. Otherwise, if the value in the data is between 3% and 15%, it means the data is medium and will be represented by a value of 2. Moreover, if the value is more than 15%, the data is indicated as poor and will be represented by a value of 1. C2 and C3 also use the same analogy as C1.

Table 7. Alternatives with criteria.

Alternative	Criteria (C)		
	Packet Loss (C1)	Delay (C2)	Throughput (C3)
A1	0	100	103
A2	100	400	105
...	...	...	...
An	100	305	190

Table 8. Criteria values.

Group	Parameter Value
Poor	1
Medium	2
Good	3

Table 9. Value of sub criteria.

No.	Criteria (C)	Range	Value
1.	C1	1. >3%	3
		2. <3–15%	2
		3. <15–25%	1
2.	C2	1. 0–150 ms	3
		2. 151–400 ms	2
		3. >400 ms	1
3.	C3	1. >25%	3
		2. 50–75%	2
		3. 100%	1

Step 2: Analysis: the weights of each criterion (packet loss, delay, and throughput) were determined using Shannon's entropy method. As the first step, rescale was done, which makes the data lie in the same range using normalization of the decision matrix (divided by the sum of each column) using Equation (7). Then, the entropy value was calculated using Equations (8) and (9). Weights vary dynamically with different data or sample sizes. The entropy method was applied to the different sizes of the data set. The results showed unobserved differences in all sample sizes, where it achieved a packet loss

weight of 0.005340, a throughput weight of 0.493564, and a delay of 0.501096 using 25% of the sample size. For the 50% and 100% sample sizes, the results are shown in Table 10.

Table 10. Criteria weights for each data sample.

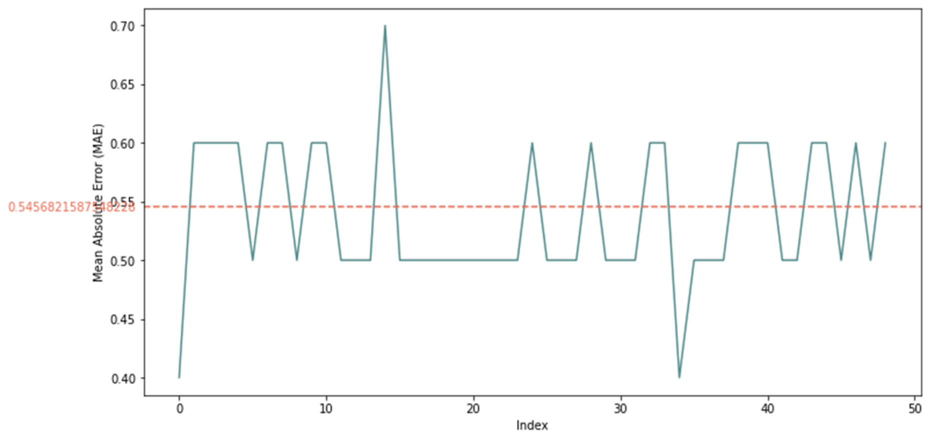
	Sample Size = 25%	
No.	**Criteria (C)**	$w_j = (\frac{\mathbf{Dj}}{\sum \mathbf{Dj}})$
1.	C1	0.005340
2.	C2	0.493564
3.	C3	0.501096
Sum		1
	Sample size = 50%	
No.	**Criteria (C)**	$w_j = (\frac{\mathbf{Dj}}{\sum \mathbf{Dj}})$
1.	C1	0.003222
2.	C2	0.498422
3.	C3	0.498355
Sum		1
	Sample size = 100%	
No.	**Criteria (C)**	$w_j = (\frac{\mathbf{Dj}}{\sum \mathbf{Dj}})$
1.	C1	0.003108
2.	C2	0.467722
3.	C3	0.529170
Sum		1

Step 3: Decision: in this step, the value of the trust was calculated and compared with DT. After obtaining the weights of the criteria values, the SMART method was used to aggregate the utility value by using Equations (12) and (13). Finally, the score of the trust was calculated using Equation (14) and compared with the threshold using Equation (15). Figure 8 shows the DT for each sample of data. Any values above the DT were considered as trusted while any values below the DT were considered as untrusted.

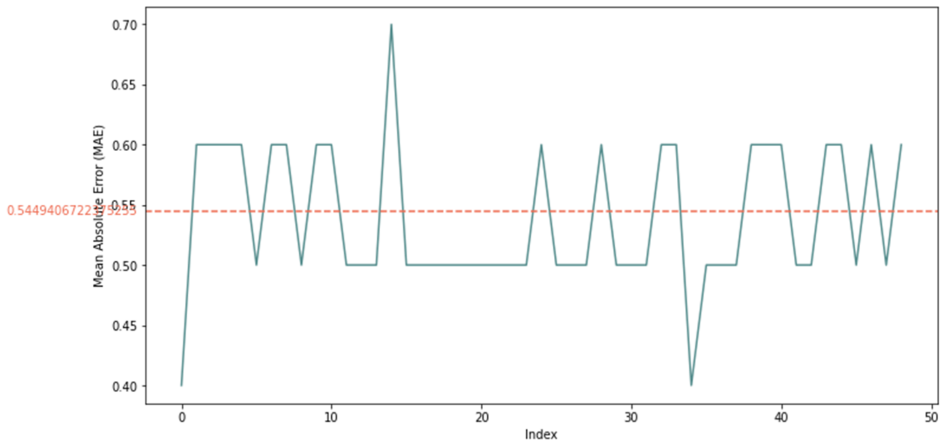
For further clarification, let us take the 100% size of the dataset (used in this study). We determined whether the device is trusted or not by investigating the three criteria (packet loss, delay, and throughput) using weighted values (0. 003108, 0.467722, and 0.529170). Let us consider an IoT device as an alternative with the following parameters: packet loss = 0%, delay = 150 ms, and throughput =75%. This means the packet loss is good, the delay is also good, and the throughput is medium. As indicated in Table 8, the values of the criteria will be 3, 3, and 2. Based on Equation (9), the values of utility are:

- The value criteria (c_{out}) = 3, then $u_i(a_i) = \frac{3-1}{3-1}$ = 1;
- The value criteria (c_{out}) = 3, then $u_i(a_i) = \frac{3-1}{3-1}$ = 1;
- The value criteria (c_{out}) = 2, then $u_i(a_i) = \frac{2-1}{3-1}$ = 0.5.

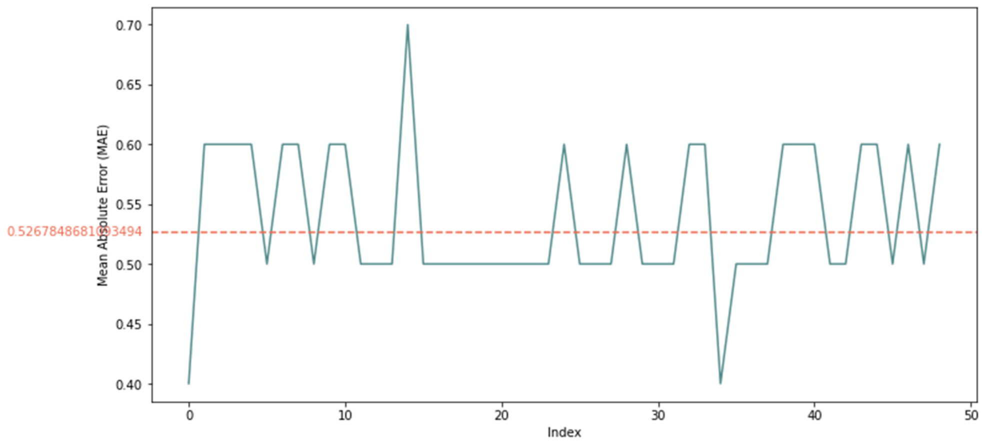
As the last step, the score was calculated using Equation (13). Therefore, the value 0.70469100 is greater than DT, which means the device is (trusted). The same mechanism was used on the whole data set to calculate the value of the trust, which was used as input for the next step (misbehaving detection).



(**a**)



(**b**)



(**c**)

Figure 8. DT results for each sample: (**a**) 25% sample size, (**b**) 50% sample size, and (**c**) 100% sample size.

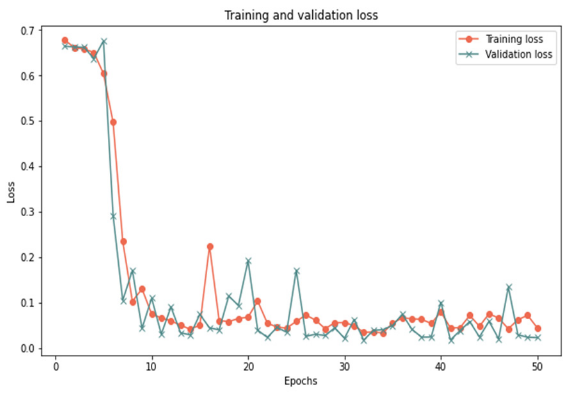
6.3. Misbehaving Detection Results

In this sub-section, the proposed model is evaluated on dataset samples of various sizes (25%, 50%, and 100%) and a different number of iterations (50 and 100) to measure the robustness of the model under various dataset settings. Table 11 reports that the proposed model for the 100% sample size shows perfect results using 50 iterations, as it obtained 99.33% accuracy, 0.023 loss rate, 100% recall, 99.26% precision, and 99.63% F-measure in 356 s. For the 100 iterations, the model shows some improvement since it obtained 99.37%

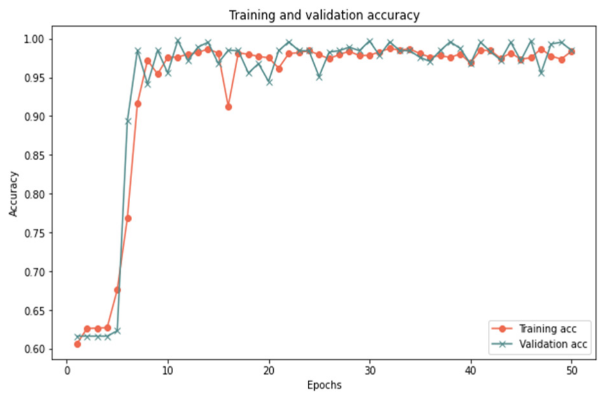
accuracy, 0.018 loss rate, 100% recall, 99.93% precision, and 99.65% F-measure in 420 s. In the 25% and 50% sample sizes, similar results are reported for 100 and 50 iterations with a slight difference in execution time. Figures 9–11 show the accuracy and the loss rate for each iteration with different samples size.

Table 11. Experimental results of the dataset with different sample sizes.

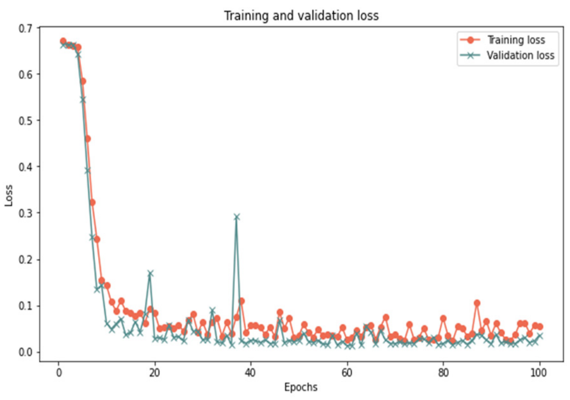
			Sample Size = 25%			
Iterations	**Accuracy (%)**	**Loss Rate**	**Recall (%)**	**Precision (%)**	**F-Measure (%)**	**Time(s)**
50	98.33	0.0350	99.85	95.97	97.87	88
100	98.50	0.00223	96.70	99.38	98.02	180
			Sample Size = 50%			
Iterations	**Accuracy (%)**	**Loss rate**	**Recall (%)**	**Precision (%)**	**F-Measure (%)**	**Time(s)**
50	98.62	0.0125	99.92	96.50	98.18	84
100	99.81	0.0115	99.92	99.69	99.81	185
			Sample Size = 100%			
Iterations	**Accuracy (%)**	**Loss rate**	**Recall (%)**	**Precision (%)**	**F-Measure (%)**	**Time(s)**
50	99.66	0.0082	98.986	100	99.49	356
100	99.97	0.0059	100	99.92	99.96	420



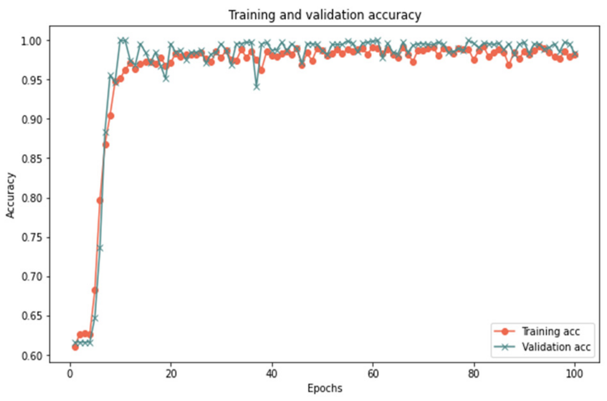
(**a**)



(**b**)

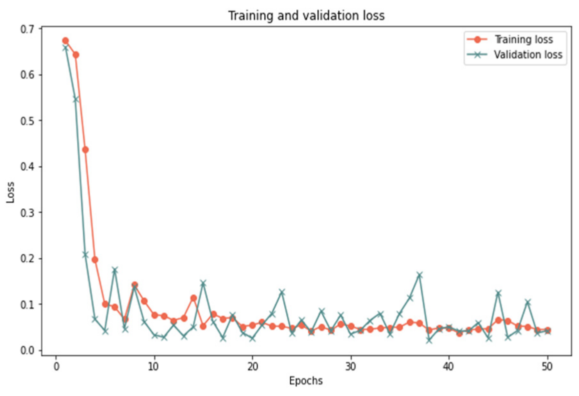


(**c**)

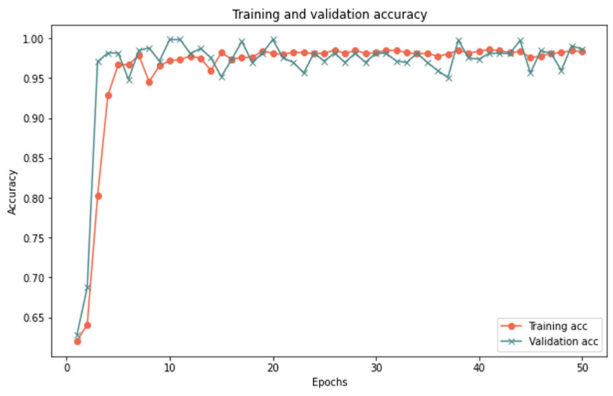


(**d**)

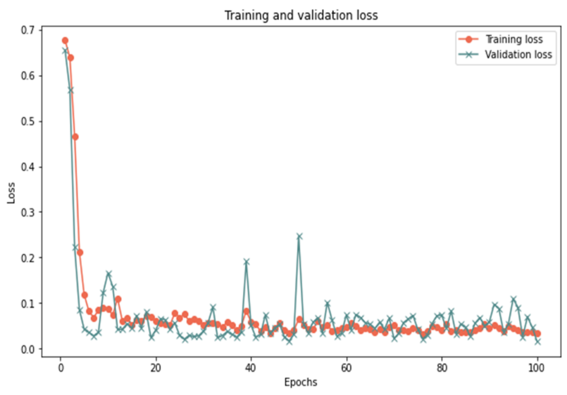
Figure 9. Results of the 25% sample size: (**a**) loss of 50 iterations, (**b**) accuracy of 50 iterations, (**c**) loss of 100 iterations, (**d**) accuracy of 100 iterations.



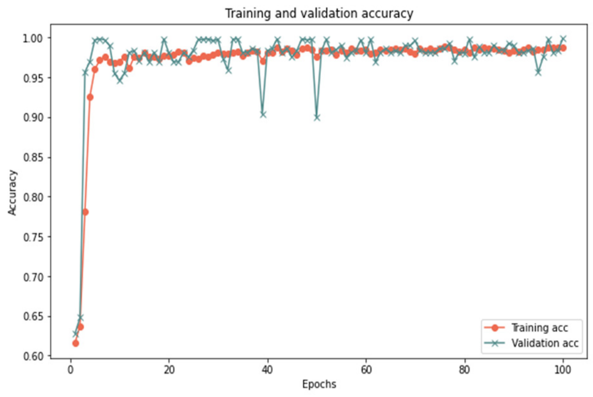
(**a**)



(**b**)



(**c**)



(**d**)

Figure 10. Results of the 50% sample size: (**a**) loss of 50 iterations, (**b**) accuracy of 50 iterations, (**c**) loss of 100 iterations, (**d**) accuracy of 100 iterations.

Based on the results presented in Sections 5.1 and 5.2, the model effectively identified behavior deviation using LSTM cells, where it produced good results for various data samples size. Changing the sample size in the experiment proved the robustness of the proposed model in dealing with different conditions. The slight difference in the results is normal and reflects the ability of the proposed model to deal with small or big data samples. The model showed an improvement in the results with the increase of the test sample. Therefore, it is clear that there is a relationship between the sample size and the model's ability to deal with big data. In addition, increasing the number of iterations has little impact on the results in terms of accuracy, recall, precision, and F-measure but a great impact on the time and loss rate. As illustrated in Figures 9–11, the results demonstrated a substantial relation between the loss rate and the number of iterations, implying that the model is learned with each iteration. Furthermore, the time and number of iterations have a positive connection; as the number of iterations increases, so does the time. There is a negative relation between accuracy, recall, precision, F-measure, and loss rates: as accuracy, recall, precision, and F-measure increase, loss rate decreases. Consequently, this proposed model can identify suspicious activities and take appropriate actions, such as helping in redirecting IoT functionality to trustworthy zones upon identifying untrusted entities.

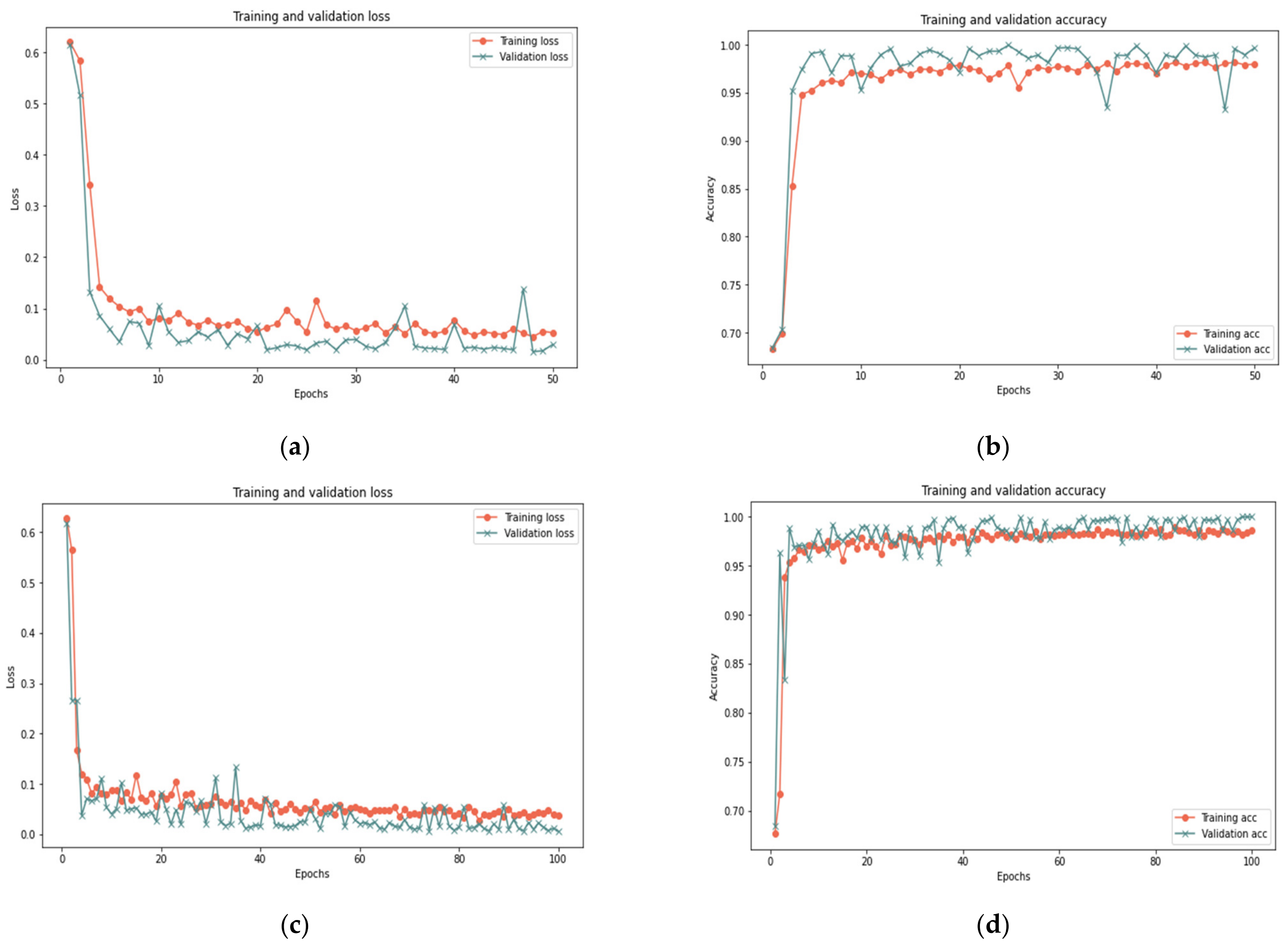


Figure 11. Results of the 100% sample size: (**a**) loss of 50 iterations, (**b**) accuracy of 50 iterations, (**c**) loss of 100 iterations, (**d**) accuracy of 100 iterations.

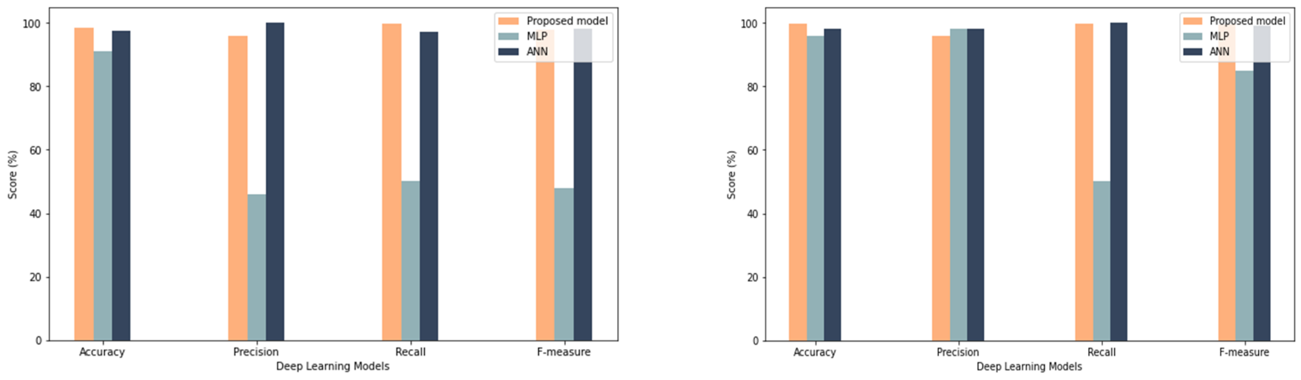
7. Comparison with Existing Approaches

Using the same dataset samples, the proposed misbehaving detection model performance is benchmarked against the most common and recent machine and deep learning-based approaches in the following subsections.

7.1. Comparison with Existing Deep Leaning Techniques

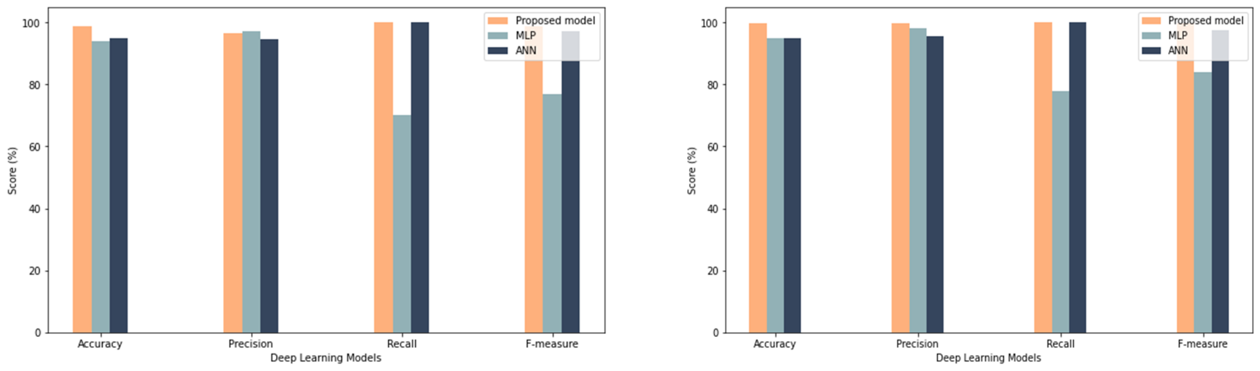
A comparison was performed between LSTM and the MLP, and the ANN, which were used in the literature as deep learning architectures (MLP in [26] and ANN in [25]) based on the model setup in Section 5.1. Theoretically, to select a suitable model, it is important to take the dataset patterns into consideration because fitting the data with the models is a condition for ensuring its effective performance. Deep learning is often used for solving big data issues, but each model is designed for specific tasks depending on the data used [31]. For example, the LSTM is usually used for time sequences and difficult learning tasks, such as prediction tasks, detecting changes in behaviors [44], machine translation [52], and handwriting generation [53]. MLP is mostly used for image processing tasks [54], and ANN is usually used for image processing, character recognition [55], and forecasting [56]. In this study, the data is IoT devices' activities, which means the type of data is behavioral patterns. Therefore, the model should be able to deal with behaviors and their changes. This proves that the LSTM model will be more compatible for identifying the changes in trusted and untrusted behaviors than other proposed models based on the ANN and MLP algorithms.

From a practical perspective, the proposed model obtained good results among all different sample sizes, which proves that the model has enhanced robustness by obtaining close results with different sample sizes, as shown in Figures 12–14. Oppositely, the ANN and MLP models showed a discrepancy between the results (such as in recall, precision, and F-measure) with the change in the size of the samples. In summary, the LSTM is an excellent choice for modeling sequential data and is thus used to understand complicated activity dynamics. The reason for this is that stacked LSTM networks may learn higher-level temporal patterns without prior knowledge of the pattern; therefore, stacked LSTM networks may be a feasible approach for modeling regular time series behavior, which can then be used to detect misbehavior.



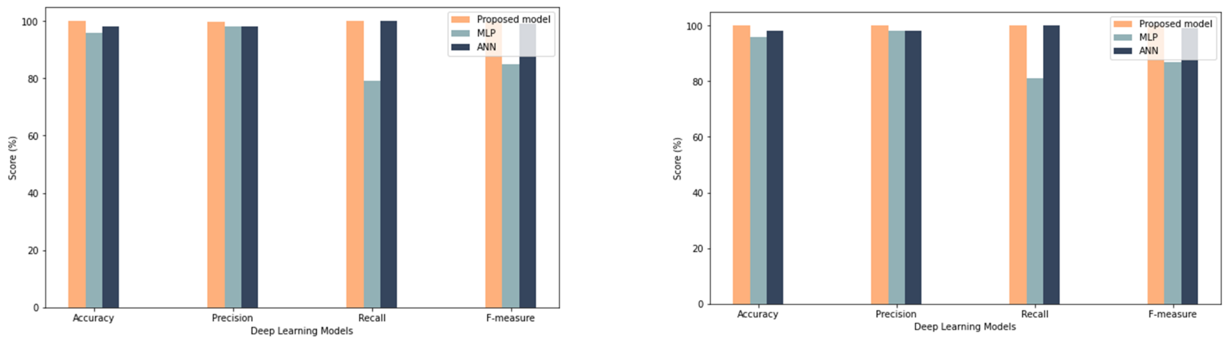
(**a**) (**b**)

Figure 12. Comparison with Deep Learning Results (**a**) shows results for 25% sample size with 50 iterations (**b**) show results for 25% sample size with 100 iterations.



(**a**) (**b**)

Figure 13. Comparison with deep learning results: (**a**) results for the 50% sample size with 50 iterations, (**b**) results for the 50% sample size with 100 iterations.

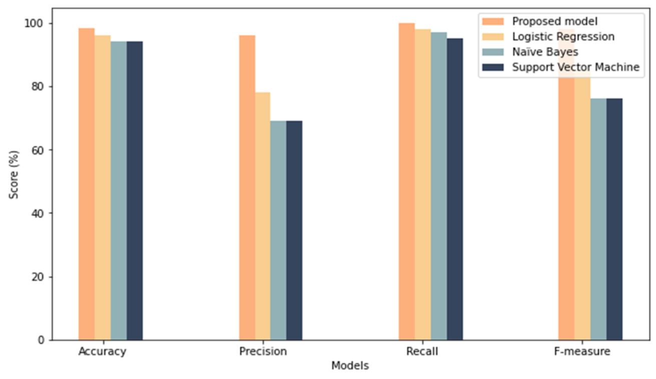


(**a**) (**b**)

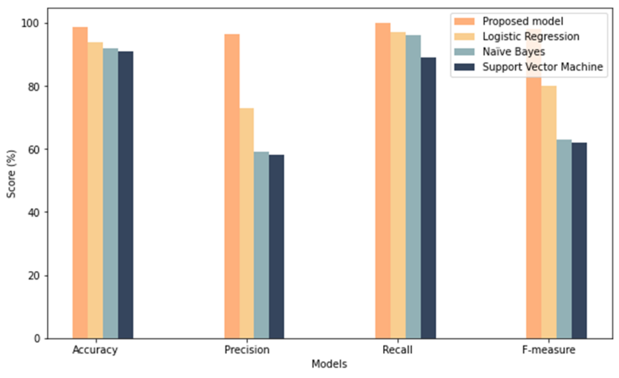
Figure 14. Comparison with deep learning results: (**a**) results for the 100% sample size with 50 iterations, (**b**) results for the 100% sample size with 100 iterations.

7.2. Comparison with Existing Machine Leaning Techniques

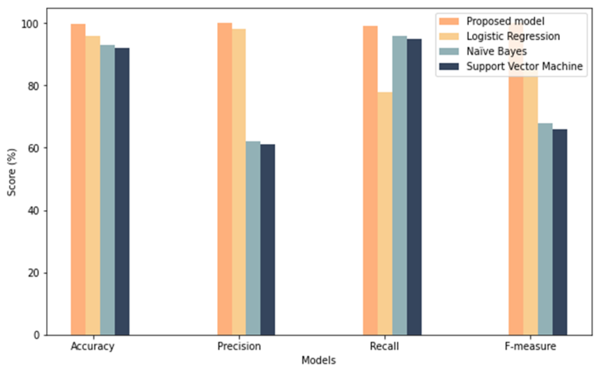
A comparison with existing machine learning techniques was performed to evaluate their performance alongside the proposed model. The comparison was performed with different samples sizes. As shown in Figure 15, the proposed model demonstrated the best overall performance among machine learning models. In particular, it showed the highest performance in terms of accuracy, recall, precision, and F-measure compared to other models. In the case of a 25% and 50% sample size, the machine learning models gave accuracy results similar to the proposed model, but in other measurements (e.g., recall, precision, and F-measure), the proposed model outperforms them. In the case of a 100% sample size, the results of the proposed model increased, while the results of the rest of the models decreased. These results gave clear evidence that the performance of the proposed model improves with the increase in the sample size used, unlike the other models whose performance decreases with the increase in the sample size. Additionally, changing the samples gave evidence that the deep learning models are more robust, as the results indicate a more stable performance despite the different samples, which makes them excellent candidates for dealing with continuous changes in IoT devices and detecting misbehavior. In conclusion, the proposed model appears to outperform other machine learning models while these models fail to reach a significant performance. This proves that the LSTM can be adapted to deal with "big data" challenges and can teach complicated behavioral patterns of IoT devices more successfully than machine learning models.



(**a**)



(**b**)



(**c**)

Figure 15. Comparison with machine learning results: (**a**) results for the 25% sample size, (**b**) results for the 50% sample size, (**c**) results for the 100% sample size.

8. Conclusions and Future Work

Managing trust is an issue with far-reaching consequences for an artificial society, such as IoT. In recent years, increased dependence on IoT devices and services has aggravated this issue. Traditional models are no longer effective in the age of big data and IoT devices have a dynamic nature and heterogeneity. As a result, this paper suggested a model for trust management in IoT devices and services that are based on the simple multi-attribute rating technique (SMART) and long short-term memory (LSTM). SMART was used to calculate the value of trust based on the information of the node itself, essentially decreasing the risk of attacks caused by incorrect recommendations. In addition, the LSTM was used to detect changes in behavior with high accuracy. Experimental results revealed that the proposed model achieved 99.87% and 99.76% accuracy and F-measure with 100 iterations, respectively. Finally, a comparison with the existing machine and deep learning techniques showed that the proposed model can achieve a superior performance in addressing trust-related problems of the IoT world. In future work, more features will be considered to calculate the value of the trust (e.g., energy consumption of IoT devices).

Author Contributions: Conceptualization, M.A.R. and Y.A.; methodology, Y.A.; software, Y.A.; validation, Y.A. and M.A.R.; formal analysis, Y.A.; investigation, M.A.R.; writing—original draft preparation, Y.A.; writing—review and editing, M.A.R.; visualization, Y.A.; supervision, M.A.R. All authors have read and agreed to the published version of the manuscript.

Funding: The researcher(s) would like to thank the Deanship of Scientific Research, Qassim University for funding the publication of this project.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Shafique, K.; Khawaja, B.A.; Sabir, F.; Qazi, S.; Mustaqim, M. Internet of Things (IoT) for Next-Generation Smart Systems: A Review of Current Challenges, Future Trends and Prospects for Emerging 5G-IoT Scenarios. *IEEE Access* **2020**, *8*, 23022–23040. [CrossRef]
2. Atzori, L.; Iera, A.; Morabito, G. The internet of things: A survey. *Comput. Netw.* **2010**, *54*, 2787–2805. [CrossRef]
3. Rajesh, G.; Raajini, X.M.; Vinayagasundaram, B. Fuzzy trust-based aggregator sensor node election in internet of things. *Int. J. Internet Protoc. Technol.* **2016**, *9*, 151. [CrossRef]
4. Hassija, V.; Chamola, V.; Saxena, V.; Jain, D.; Goyal, P.; Sikdar, B. A Survey on IoT Security: Application Areas, Security Threats, and Solution Architectures. *IEEE Access* **2019**, *7*, 82721–82743. [CrossRef]
5. Fortino, G.; Russo, W.; Savaglio, C.; Viroli, M.; Zhou, M. Opportunistic cyberphysical services: A novel paradigm for the future Internet of Things. In Proceedings of the 2018 IEEE 4th World Forum on Internet of Things (WF-IoT), Singapore, 7 May 2018; pp. 488–492.
6. Sfar, A.R.; Natalizio, E.; Challal, Y.; Chtourou, Z. A roadmap for security challenges in the Internet of Things. *Digit. Commun. Netw.* **2018**, *4*, 118–137. [CrossRef]
7. Jayasinghe, U.; Lee, G.M.; Um, T.-W.; Shi, Q. Machine Learning Based Trust Computational Model for IoT Services. *IEEE Trans. Sustain. Comput.* **2019**, *4*, 39–52. [CrossRef]
8. Najib, W.; Sulistyo, S. Widyawan Survey on Trust Calculation Methods in Internet of Things. *Procedia Comput. Sci.* **2019**, *161*, 1300–1307. [CrossRef]
9. Yan, Z.; Zhang, P.; Vasilakos, A.V. A survey on trust management for Internet of Things. *J. Netw. Comput. Appl.* **2014**, *42*, 120–134. [CrossRef]
10. Djedjig, N.; Tandjaoui, D.; Romdhani, I.; Medjek, F. Trust management in the internet of things. In *Security and Privacy in Smart Sensor Networks*; IGI Global: Hershey, PA, USA, 2018; pp. 122–146.
11. Khalil, A.; Mbarek, N.; Togni, O. Fuzzy Logic Based Security Trust Evaluation for IoT Environments. In Proceedings of the 2019 IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA), Abu Dhabi, United Arab Emirates, 3–7 November 2019; pp. 1–8.
12. Asiri, S.; Miri, A. An IoT trust and reputation model based on recommender systems. In Proceedings of the 2016 14th Annual Conference on Privacy, Security and Trust (PST), Auckland, New Zealand, 12–14 December 2016; pp. 561–568.
13. Mendoza, C.V.L.; Kleinschmidt, J.H. Mitigating On-Off Attacks in the Internet of Things Using a Distributed Trust Management Scheme. *Int. J. Distrib. Sens. Netw.* **2015**, *11*, 859731. [CrossRef]
14. Che, S.; Feng, R.; Liang, X.; Wang, X. A lightweight trust management based on Bayesian and Entropy for wireless sensor networks. *Secur. Commun. Netw.* **2015**, *8*, 168–175. [CrossRef]
15. Ahmed, A.I.A.; Ab Hamid, S.H.; Gani, A.; Khan, S.; Khan, M.K. Trust and reputation for Internet of Things: Fundamentals, taxonomy, and open research challenges. *J. Netw. Comput. Appl.* **2019**, *145*, 102409. [CrossRef]
16. Bao, F.; Chen, I.-R.; Guo, J. Scalable, adaptive and survivable trust management for community of interest based Internet of Things systems. In Proceedings of the 2013 IEEE Eleventh International Symposium on Autonomous Decentralized Systems (ISADS), Mexico City, Mexico, 6–8 March 2013; pp. 1–7.

17. Nitti, M.; Girau, R.; Atzori, L. Trustworthiness Management in the Social Internet of Things. *IEEE Trans. Knowl. Data Eng.* **2014**, *26*, 1253–1266. [CrossRef]
18. Chen, I.-R.; Guo, J.; Bao, F. Trust Management for SOA-Based IoT and Its Application to Service Composition. *IEEE Trans. Serv. Comput.* **2016**, *9*, 482–495. [CrossRef]
19. Renubala, S.; Dhanalakshmi, K.S. Trust based secure routing protocol using fuzzy logic in wireless sensor networks. In Proceedings of the 2014 IEEE International Conference on Computational Intelligence and Computing Research, Coimbatore, India, 18–20 December 2014; pp. 1–5.
20. Alshehri, M.D.; Hussain, F.K. A centralized trust management mechanism for the internet of things (ctm-iot). In *International Conference on Broadband and Wireless Computing, Communication and Applications*; Springer: New York, NY, USA, 2017.
21. Alshehri, M.D.; Hussain, F.K.; Hussain, O.K. Clustering-driven intelligent trust management methodology for the internet of things (CITM-IoT). *Mob. Netw. Appl.* **2018**, *23*, 419–431. [CrossRef]
22. Alshehri, M.D.; Hussain, F.K. A fuzzy security protocol for trust management in the internet of things (Fuzzy-IoT). *Computing* **2019**, *101*, 791–818. [CrossRef]
23. Masmoudi, M.; Abdelghani, W.; Amous, I.; Sèdes, F. Deep Learning for Trust-Related Attacks Detection in Social Internet of Things. In *Evolutionary Computing and Mobile Sustainable Networks*; Springer: Singapore, 2020; pp. 389–404.
24. Wang, B.; Li, M.; Jin, X.; Guo, C. A Reliable IoT Edge Computing Trust Management Mechanism for Smart Cities. *IEEE Access* **2020**, *8*, 46373–46399. [CrossRef]
25. Ba-Hutair, M.N.; Bouguettaya, A.; Neiat, A.G. Multi-Perspective Trust Management Framework for Crowdsourced IoT Services. *IEEE Trans. Serv. Comput.* **2021**, 1. [CrossRef]
26. Hankare, P.; Babar, S.; Mahalle, P. Trust Management Approach for Detection of Malicious Devices in SIoT. *Teh. Glas.* **2021**, *15*, 43–50. [CrossRef]
27. Zheng, G.; Gong, B.; Zhang, Y. Dynamic Network Security Mechanism Based on Trust Management in Wireless Sensor Networks. *Wirel. Commun. Mob. Comput.* **2021**, *2021*, 6667100. [CrossRef]
28. Lingda, K.; Feng, Z.; Yingjie, Z.; Nan, Q.; Dashuai, L.; Shaotang, C. Evaluation method of trust degree of distribution IoT terminal equipment based on information entropy. *J. Phys. Conf. Ser.* **2021**, *1754*, 012108. [CrossRef]
29. Din, I.U.; Guizani, M.; Kim, B.-S.; Hassan, S.; Khan, M.K. Trust Management Techniques for the Internet of Things: A Survey. *IEEE Access* **2018**, *7*, 29763–29787. [CrossRef]
30. Pourghebleh, B.; Navimipour, N.J. Data aggregation mechanisms in the Internet of things: A systematic review of the literature and recommendations for future research. *J. Netw. Comput. Appl.* **2017**, *97*, 23–34. [CrossRef]
31. Yue, Y.; Li, S.; Legg, P.; Li, F. Deep Learning-Based Security Behaviour Analysis in IoT Environments: A Survey. *Secur. Commun. Netw.* **2021**, *2021*, 8873195. [CrossRef]
32. Anagnostopoulos, M.; Spathoulas, G.; Viaño, B.; Augusto-Gonzalez, J. Tracing Your Smart-Home Devices Conversations: A Real World IoT Traffic Data-Set. *Sensors* **2020**, *20*, 6600. [CrossRef] [PubMed]
33. Crawford, M.; Khoshgoftaar, T.M.; Prusa, J.D.; Richter, A.N.; Al Najada, H. Survey of review spam detection using machine learning techniques. *J. Big Data* **2015**, *2*, 23. [CrossRef]
34. Sugeng, W.; Istiyanto, J.E.; Mustofa, K.; Ashari, A. The impact of QoS changes towards network performance. *Int. J. Comput. Netw. Commun. Secur.* **2015**, *3*, 48–53.
35. Zach. *Normailzation in Statology 2021*; Statology: Torrance, CA, USA, 2021.
36. Oktavianti, E.; Komala, N.; Nugrahani, F. Simple multi attribute rating technique (SMART) method on employee promotions. *J. Phys. Conf. Ser.* **2019**, *1193*, 012028. [CrossRef]
37. Risawandi, R.R.; Rahim, R. Study of the simple multi-attribute rating technique for decision support. *Decis. -Mak.* **2016**, *4*, C4.
38. Işık, A.T.; Adalı, E.A. The Decision-Making Approach Based on the Combination of Entropy and Rov Methods for the Apple Selection Problem. *Eur. J. Interdiscip. Stud.* **2017**, *8*, 80–86. [CrossRef]
39. Jati, H.; Dominic, D.D. A New Approach of Indonesian University Webometrics Ranking Using Entropy and PROMETHEE II. *Procedia Comput. Sci.* **2017**, *124*, 444–451. [CrossRef]
40. Lotfi, F.H.; Fallahnejad, R. Imprecise Shannon's Entropy and Multi Attribute Decision Making. *Entropy* **2010**, *12*, 53–62. [CrossRef]
41. Reich, N.G.; Lessler, J.; Sakrejda, K.; Lauer, S.A.; Iamsirithaworn, S.; Cummings, D.A.T. Case Study in Evaluating Time Series Prediction Models Using the Relative Mean Absolute Error. *Am. Stat.* **2016**, *70*, 285–292. [CrossRef]
42. Khani, M.; Wang, Y.; Orgun, M.A.; Zhu, F. Context-aware trustworthy service evaluation in social internet of things. In Proceedings of the International Conference on Service-Oriented Computing, Hangzhou, China, 12–15 November 2018.
43. Chen, Z.; Ling, R.; Huang, C.-M.; Zhu, X. A scheme of access service recommendation for the Social Internet of Things. *Int. J. Commun. Syst.* **2016**, *29*, 694–706. [CrossRef]
44. Mekruksavanich, S.; Jitpattanakul, A. Biometric User Identification Based on Human Activity Recognition Using Wearable Sensors: An Experiment Using Deep Learning Models. *Electronics* **2021**, *10*, 308. [CrossRef]
45. Alghofaili, Y.; Albattah, A.; Rassam, M.A. A Financial Fraud Detection Model Based on LSTM Deep Learning Technique. *J. Appl. Secur. Res.* **2020**, *15*, 498–516. [CrossRef]
46. Zhao, Z.; Xu, C.; Li, B. A LSTM-Based Anomaly Detection Model for Log Analysis. *J. Signal Process. Syst.* **2021**, *93*, 745–751. [CrossRef]

47. Kim, T.-Y.; Cho, S.-B. Web traffic anomaly detection using C-LSTM neural networks. *Expert Syst. Appl.* **2018**, *106*, 66–76. [CrossRef]
48. Chicco, D.; Jurman, G. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genom.* **2020**, *21*, 6. [CrossRef]
49. Tharwat, A. Classification assessment methods. *Appl. Comput. Inform.* **2020**, *17*, 168–192. [CrossRef]
50. Dalianis, H. Evaluation metrics and evaluation. In *Clinical Text Mining*; Springer: New York, NY, USA, 2018; pp. 45–53.
51. Fayyaz, Z.; Ebrahimian, M.; Nawara, D.; Ibrahim, A.; Kashef, R. Recommendation Systems: Algorithms, Challenges, Metrics, and Business Opportunities. *Appl. Sci.* **2020**, *10*, 7748. [CrossRef]
52. Sutskever, I.; Vinyals, O.; Le, Q.V. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems 2014*; Curran Associates, Inc.: Red Hook, NY, USA, 2014; Volume 27, pp. 3104–3112.
53. Graves, A. Generating Sequences with Recurrent Neural Networks. *arXiv* **2013**, arXiv:1308.0850.
54. Shawky, O.A.; Hagag, A.; El-Dahshan, E.-S.A.; Ismail, M.A. Remote sensing image scene classification using CNN-MLP with data augmentation. *Optik* **2020**, *221*, 165356. [CrossRef]
55. Mehdy, M.M.; Ng, P.Y.; Shair, E.F.; Saleh, N.I.M.; Gomes, C. Artificial Neural Networks in Image Processing for Early Detection of Breast Cancer. *Comput. Math. Methods Med.* **2017**, *2017*, 2610628. [CrossRef] [PubMed]
56. Notton, G.; Voyant, C.; Fouilloy, A.; Duchaud, J.L.; Nivet, M.L. Some Applications of ANN to Solar Radiation Estimation and Forecasting for Energy Applications. *Appl. Sci.* **2019**, *9*, 209. [CrossRef]