applied sciences

MDPI

Article

Real-Time Anomaly Detection with Subspace Periodic Clustering Approach

Fokrul Alom Mazarbhuiya [1,*] and Mohamed Shenify [2,*]

[1] School of Fundamental and Applied Sciences, Assam Don Bosco University, Guwahati 782402, India
[2] College of Computer Science and IT, Al-Baha University, Al-Baha 65799, Saudi Arabia
* Correspondence: fokrul.mazarbhuiya@dbuniversity.ac.in (F.A.M.); maalshenify@bu.edu.sa (M.S.)

Abstract: Finding real-time anomalies in any network system is recognized as one of the most challenging studies in the field of information security. It has so many applications, such as IoT and Stock Markets. In any IoT system, the data generated is real-time and temporal in nature. Due to the extreme exposure to the Internet and interconnectivity of the devices, such systems often face problems such as fraud, anomalies, intrusions, etc. Discovering anomalies in such a domain can be interesting. Clustering and rough set theory have been tried in many cases. Considering the time stamp associated with the data, time-dependent patterns including periodic clusters can be generated, which could be helpful for the efficient detection of anomalies by providing a more in-depth analysis of the system. Another issue related to the aforesaid data is its high dimensionality. In this paper, all the issues related to anomaly detection are addressed, and a clustering-based approach is proposed for finding real-time anomalies. The method employs rough set theory, a dynamic *k*-means clustering algorithm, and an interval superimposition approach for finding periodic, partially periodic, and fuzzy periodic clusters in the subspace of the dataset. The data instances are thought to be anomalous if they either belong to sparse clusters or do not belong to any clusters. The efficacy of the method can be assessed by means of both time-complexity analysis and comparative studies with existing clustering-based anomaly detection algorithms on a synthetic and a real-life dataset. It can be found experimentally that our method outperforms others and runs in cubic time.

Keywords: anomaly detection; information system; high-dimensional data; dominance relation; *k*-means algorithm; fuzzy periodic cluster

check for updates

Citation: Mazarbhuiya, F.A.; Shenify, M. Real-Time Anomaly Detection with Subspace Periodic Clustering Approach. *Appl. Sci.* **2023**, *13*, 7382. https://doi.org/10.3390/app13137382

Academic Editor: Dimitris Mourtzis

Received: 22 May 2023
Revised: 19 June 2023
Accepted: 20 June 2023
Published: 21 June 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

1.1. Origin of the Problem

Due to the extensive use of computers, networks, and databases, they are exposed to various types of attacks. The attacks may be in the form of hacking, intruding, etc. The term "anomalous activity" has been used for this, and any data instance associated with such attacks is known as "anomaly". Finding such anomalies is one of the hot areas of research in modern times. It has so many applications, such as IoT, the stock market, banking, security, defense, etc. Over the past few years, Internet of Things (IoT) networks have brought significant changes to individual lives, society, and industry [1,2]. The IoT devices consist of a huge number of sensors generating data over time [3], and as a result, the availability of streaming time-series data is expanding exponentially. As a result of the involvement of a wide range of information and communication technologies, IoT networks are exposed to various types of security threats [4,5]. In other words, any system that relies on the IoT faces huge security and privacy challenges [6,7]. The challenges are in the form of anomalies, intrusions, or any other illegitimate activities that jeopardize the security of the system [8]. Although the system can be protected to some extent by the defense mechanisms currently in place, malicious attackers are becoming more skilled at breaking networks. Again, in an event such as an inside attack, it is more challenging to

prevent it in real-time. Therefore, identifying such real-time attacks can provide actionable information in dire situations for which there are no trustworthy solutions [9–18]. Here, a new and reliable clustering-based method is put forth to address the problem.

Unsupervised learning techniques such as clustering [19] are widely used to determine the distribution of data and patterns. It has recently been employed in anomaly detection as well as other branches such as psychology and social science, where it has long been extensively used [20,21]. Static clustering and dynamic clustering are the two primary categories of clustering techniques. Static clustering primarily targets static datasets that are prepared before the algorithm is applied. Dynamic clustering is necessary in some applications using real-time data, such as cloud computing, IoT, finance, and stock markets. A hierarchical approach that may be applied to both static and dynamic datasets was proposed by the authors in [22]. Several incremental clustering algorithms were put forth by the authors in [23] in order to process new records or data instances as and when they are added.

There are mainly two problems encountered while dealing with anomalies in any IoT-based system: the high dimensionality of the data and the real-time detection of anomalies. Anomalies are often hard to find at high dimensionality. For that reason, more data are necessary to properly generalize as the number of attributes or features rises, which results in data sparsity. Data sparsity is brought on by these additional attributes or a sizable amount of noise from several irrelevant attributes that obscure the real anomalies. The term "curse of dimensionality" [24,25] is a well-known one used to describe the issue. As a result, it has been discovered that numerous traditional anomaly detection methods [26–28] are inappropriate for high-dimensional data because they lose their effectiveness. In [29], the authors suggested a method for high-dimensional and categorical data anomaly detection. Similar works were presented in [30–33]. Again, any IoT system generates real-time data. In such systems, anomalies can be temporal or contextual [34] in nature, where the temporal order of the data is significant. To put it another way, a data instance can only be anomalous within a particular temporal context, or more accurately, within a timeframe. Some of these anomalies can be periodic in nature, i.e., they occur after a certain interval of time. These anomalies are difficult to detect. The development of an early warning system is one of the key applications of such abnormalities. In view of the above scenario, it is necessary to design an effective algorithm that locates clusters in the subspace of high-dimensional real-time data and detects anomalies in real-time.

Pawlak proposed rough set theory [35] to address the ambiguity and uncertainty that can be found in any dataset. Thivagar et al. [36,37] gave the notion of nano topological space in terms of the two approximations and generated CORE, a subset of conditional attributes used for medical diagnosis. The same notion can be used for generating a subset of high-dimensional real-time data. A comparison of five time-series anomaly detection techniques was conducted by the authors of [38]. Similar efforts were mentioned in [39–47]. The insider threat, which creates significant issues for the cyber security of industrial control systems, was addressed by the authors of [48]. A random forest-based strategy for online anomaly detection was presented by Zhao et al. [49]. In [50–52], the authors offered fuzzy-based approaches for real-time anomaly detection. In [53], the authors suggested a fuzzy neural network approach with the goal of identifying anomalies in significant cyberattacks. An effective real-time clustering-based anomaly detection system was described by the authors in [54].

As mentioned earlier, while detecting anomalies in most cases, two problems are frequently encountered, namely the high dimensionality of the data and the temporal reference of the detected anomaly. High dimensionality is a serious issue that reduces the efficacy of any anomaly detection method. Also, the temporal reference of the anomaly is important as it gives the time of occurrence of the anomalous data, which can be useful for the design of an early warning system. Though several methods tried to address the aforesaid problems, only a few successfully addressed one or another. So, there is enough scope to work on these problems.

1.2. Motivation and Contribution

Most of the algorithms discussed above have some limitations. For example, some are inefficient at finding anomalies in high-dimensional data, and others are unable to find real-time anomalies. Although there exist many algorithms [7–9,15,18,29,34,38,42,43] for efficient detection of real-time anomalies, only a few actually address the periodicity in the real-time data. In real-time data such as sensors, IT monitoring applications generate huge volume of data continuously over a period of time, which is the lifespan of the dataset. Over the lifespan of the dataset, there may be data instances of similar nature that occur periodically. Considering the time attribute associated with such data as calendar dates (year_month_day_hour_minute_second), periodic clusters can be generated where the period of a cluster can be represented as a sequence of lifespans of clusters. In other words, in such a system, it would be interesting to observe whether the clusters or anomalies generated are of a periodic nature or not. This information can be useful for predicting anomalies. In [55], the authors proposed calendar-based periodic patterns from supermarket datasets. With the help of an interval superimposition operation [56–58], the algorithm finds a match ratio to generate fully, partially, and fuzzy [59] periodic patterns.

In this article, the problems of high-dimensionality, real-time detection of anomalies, and periodicity have been efficiently addressed—and a method is proposed that can generate fully periodic, partially periodic, and fuzzy periodic clusters. It is named the RADSPCA. The method uses the notion of rough set theory and the *k*-means clustering algorithm to generate clusters along with their sequence of lifespans, and then the interval superimposition is applied to the lifespans to generate the periodicity of the clusters. The objective of the paper is as follows:

Firstly, a dominance relation is defined on the dataset [60].

Secondly, an interval superimposition operation is defined, and a match ratio in terms of interval superimposition is also defined.

Finally, a new clustering-based method is proposed to generate periodic, partially periodic, and fuzzy periodic clusters in the subspace of the dataset.

Thus, the RADSPCA first uses a rough set theoretic approach to find a lower-dimensional space by removing the irrelevant attributes. Then, the dynamic *k*-means clustering algorithm is applied to it to find the clusters along with their list of lifespans. At the end of this stage, each cluster will have a list of lifespans describing its period. Then the interval superimposition operation is applied to the list of lifespans to generate superimposed time-intervals along with their match ratio [55,56]. The match ratio will determine whether the corresponding cluster is fully or partially periodic. Also, by applying a nice method [56,57] on superimposed intervals, fuzzy time intervals can be generated. This way, from each fully or partially periodic cluster fuzzy periodic clusters can be generated. Then, RADSPCA's complexity is estimated. Lastly, a detailed comparative analysis is conducted with existing well-known clustering-based methods [9,10,19,46,54,61–63] using a MATLAB implementation with first KDDCUP'99 [64] and then Kitsune Network Attack [65] datasets. The results effectively validate our technique.

The structure of the paper is as follows: In Section 2, it is discussed how this field has recently advanced. In Section 3, the problem definition is presented. Section 4 covers the proposal method (RADSPCA). Section 5 discusses the time-complexity. Section 6 of the paper contains the experiments, results, and analysis, and Section 7 of the paper contains the conclusions, limitations, and future directions.

2. Related Works

Anomaly detection is the search for patterns that differ from previously known ones. It is useful for obtaining sufficient information about the system that generates, processes, or transmits the data. Since the last couple of years, sensors and application-based IoT networks have become popular to substantially upgrade the standard of individual life by contributing to the development of society and industry [1,2]. Such systems generate data exponentially over time [3]. However, due to the involvement of the Internet and other

communication technologies, the networks are always open to various security threats [4–6]. So, for such a system, the security and privacy challenges [7] are major causes of concern. Some of the common challenges are anomalies, fraud, intrusion, or any other illegitimate activities that jeopardize the system's reliability [8]. Currently, the protection of networks exists to a reasonable extent, but malicious attackers are becoming smart enough to break into the networks again and again. In [9], the authors proposed a hybrid approach using both partitioning and an agglomerative hierarchical clustering algorithm for real-time anomaly detection. Using a unified metric [9] defined on both numeric and categorical attributes, a distance function and similarity measure are expressed which are then used for generating clusters on fuzzy time intervals. However, the above method did not supply periodic clusters and periodic anomalies. In [10], the author used a merge function in the *k*-means algorithm to generate anomalies from a mixed attribute dataset. Though the work of [10] is quite similar to [9], the later method supplies real-time anomalies, which the former does not. In [11], the authors put forward an agglomerative hierarchical model for the detection of anomalies in a network dataset. [12] built a rough set-based classification model for anomaly detection. Applying automatic labeling for supervised learning, an anomaly detection scheme was proposed in [13]. In [14], an unsupervised deep learning approach was proposed that can detect the potential correlation features among multi-dimensional sensor data and find out the anomalies in public transportation and facilities in smart cities. The work of [14] efficiently addressed some of the issues, such as the multi-dimensionality and the periodicity of the real-life data. However, the periodicity of clusters or anomalies was not addressed. In [15], the authors offered both semi-supervised and supervised approaches for real-time anomaly detection in a high-dimensional data stream. In [16], the authors used correlation laws to detect anomalies. In [17], the authors proposed a new method incorporating neural processes into a semi-supervised anomaly detection model. In [18], the authors conducted a detailed review on anomaly detection paradigms based on offline-learning, semi-online learning and online-learning. in high-velocity data streams.

Cheng et al. [20], proposed a unified metric defined by mixed attributes to generate clusters. In [21], the authors offered an agglomerative hierarchical model for clustering periodic patterns, where a pattern is said to be periodic if it occurs repeatedly after a certain period of time. There are mainly two problems encountered while dealing with anomalies in any real-time data, namely, the high dimensionality and the real-time detection of anomalies. In [24], the authors tried to address the 'curse of dimensionality issue effectively and used a one-class support vector machine for the effective detection of anomalies from high-dimensional data. In [25], the authors introduced a survey on contemporary anomaly detection paradigms. Kaya et al. [26] analyzed different methodologies for communication pattern recognition. Considering the fact that the sparsity of data in high-dimensional spaces leads to the failure of the effectiveness of any anomaly detection method in [27], the authors suggested an efficient scheme for detecting high-dimensional anomalies. In [28] the authors addressed the high dimensionality and proposed an unsupervised method for anomaly detection in such data. In [32], the authors presented a hybrid approach consisting of a semi-supervised approach for anomaly detection in high-dimensional data. In [33], the authors proposed a mixed approach consisting of rough set theory and a density-based clustering algorithm for anomaly detection in high-dimensional data. In [34], the authors addressed the issue of the temporality of anomalies and proposed a clustering-based system for real-time anomaly detection in streaming data. Most of the aforesaid methods tried to address high dimensionality efficiently but the periodicity of clusters or anomalies was not discussed.

Rough set theory as a tool to deal with the ambiguity and uncertainty occurring in any real system was proposed by Pawlak [35]. In [36], the authors applied the rough set theory to produce nano topology. In [37], the authors applied the notion of nano topology for medical diagnosis. The same notion can be used for attribute reduction in high-dimensional data [33]. Halstead et al. [44], proposed a method using diverse meta-features

for identifying recurring concepts of drift in data streams. In [45], the authors put forward a two-layered classification model for the online anomaly detection of highly unreliable data. In [46], the authors presented a scheme for the online detection of anomalies in data streams. In [47], the authors proposed to evaluate cyber risk for operation technology systems. In [48], the authors discussed insider threat, which creates significant issues for the cyber security of industrial control systems. Zhao et al. [49] presented an online anomaly detection model based on a random forest method. Izakian et al. [50] proposed to introduce fuzzy in anomaly detection by proposing a fuzzy c-means-based technique. Souza et al. [53] presented a fuzzy neural network-based approach for detecting anomalies in massive cyberattacks. In [54], the authors presented an effective clustering-based real-time anomaly detection system. Mahanta et al. [55] proposed a calendar-based periodic pattern from supermarket datasets. In [56], the authors used an interval operation called interval superimposition to find the solution of a fuzzy linear equation. In [58], the authors proposed a lemma called the Glivenko–Cantelli lemma. Using the lemma on superimposed intervals [56], fuzzy intervals [57,59] can be generated. In [60], the authors proposed a dominance relation on conditional attributes to generate set-valued ordered information systems which can be used for attribute reduction in the dataset. In most of the anomaly detection cases discussed above, the methods tried to propose effective ways to deal with either high-dimensionality or real-time issues. However, a few were able to deal with both effectively. Also, no algorithms have addressed the periodicity of clusters or anomalies in the datasets. However, the periodicity of the clusters or anomalies can be very useful in designing early warning systems. In this article, we propose to address all the aforesaid issues effectively and present a suitable solution for them.

3. Problem Definitions

In below, we present some important terms and definitions used in this paper.

Definition 1. *Let U be a non-empty finite set of objects, A, a finite set of attributes, and* $V = \cup V_a$, *where* V_a *is the domain of the attribute* $a \in A$, *then quadruple* $S = (U, A, V, f)$ *defines a set-valued information system [60]. A function* $f: U \times A \rightarrow V$ *is defined as* $1 \leq f(x, a) \in V_a, \forall x \in U$, $a \in A$. *Also, we take the attribute set* $A = \{C \cup \{d\}; C \cap \{d\} = \phi; C$, *the set of conditional and* $\{d\}$ *the decision attributes}.*

Definition 2. *If the domain of a conditional attribute in C can be arranged in ascending or descending order of preference, then such attribute is called as criterion [36]. If every conditional attribute of C is a criterion, then the information system is known as the set-valued ordered information system [60].*

Definition 3. *The attribute is an inclusion criterion if the values of some objects in U under a conditional attribute of C can be sorted according to inclusion increasing or decreasing preferences [60].*

Definition 4. *Let us define a relation* $\boldsymbol{R}_{\boldsymbol{A}}^{\geq}$ on a set-valued ordered information system *[60] with inclusion increasing preference as, [see e.g., [46]]*

$$R_A^{\geq} = \{(y, x) \in U \times U : f(y, a) \geq f(x, a) \forall a \in A\} \tag{1}$$

then $R_A^{\geq}$ *is said to be the dominance relation on U. When* $(y, x) \in R_A^{\geq}$ *then* $y \geq_A x$ *, that means y is at least as good as x with reference to A.*

Property 1. *The inclusion dominance relation* $\boldsymbol{R}_{\boldsymbol{A}}^{\geq}$ *[60] is (i) reflexive, (ii) unsymmetric, and (iii) transitive.*

Definition 5. *For* $x \in U$, *the dominance class [36,37] of x is defined as*

$$[x]_A^{\geq} = \left\{ y \epsilon U : (y, x) \epsilon R_A^{\geq} \right\} = \{ y \epsilon U : f(y, a) \geq f(x, a), \forall a \epsilon U \} \tag{2}$$

where $U/_A^{\geq} = \left\{ [x]_A^{\geq} : x \epsilon U \right\}$ *is the family of dominance classes.*

Remark 1. $\boldsymbol{U/_A^{\geq}}$ *is not a partition of U, rather it creates a covering of U, that is* $U = \cup\, [\boldsymbol{x}]_{\boldsymbol{A}}^{\geq}$.

Definition 6. *For any set-valued ordered information system S = {U, A, V, f} and for a given* $X \subseteq U$*, the upper approximation, lower approximation and boundary regions of X are respectively expressed as [36,37]*

$$\boldsymbol{U_A^{\geq}(X)} = \left\{ \boldsymbol{x \epsilon U} : [\boldsymbol{x}]_{\boldsymbol{A}}^{\geq} \cap \boldsymbol{X} \neq \boldsymbol{\phi} \right\} \tag{3}$$

$$\boldsymbol{L_A^{\geq}(X)} = \left\{ \boldsymbol{x \epsilon U} : \left[\boldsymbol{x}\right]_{\boldsymbol{A}}^{\geq} \subseteq \boldsymbol{X} \right\} \tag{4}$$

$$\boldsymbol{B_A^{\geq}(X)} = \boldsymbol{U_A^{\geq}(X)} - \boldsymbol{L_A^{\geq}(X)} \tag{5}$$

Definition 7. *For a set-valued ordered information system, S = {U, A, V, f}, B (*$\subseteq A$*) is termed as criterion reduction of S if* $\boldsymbol{R_A^{\geq}} = \boldsymbol{R_B^{\geq}}$ *and* $\boldsymbol{R_M^{\geq}} \neq \boldsymbol{R_A^{\geq}}$ *for any* $M \subseteq A$. On the other way, *a minimal attribute set B is a criterion reduction of S if* $\boldsymbol{R_A^{\geq}} = \boldsymbol{R_B^{\geq}}$ *[36,37].*

Definition 8. *A CORE is a minimal subset of attributes such that none of its elements can be removed without violating the nano topology generated by the lower and upper approximations. The CORE(A) is given by CORE(A) =* $\left\{ \boldsymbol{a \epsilon A} : \boldsymbol{R_A^{\geq}} \neq \boldsymbol{R_{A-\{a\}}^{\geq}} \right\}$ *[see e.g., [36,37]].*

Definition 9. *Let* $\boldsymbol{R_C^{\geq}}$ *be a dominance relation on U, then* $\boldsymbol{\tau_C^{\geq}(X)} = \left\{ \boldsymbol{U, \phi, U_C^{\geq}(X), L_C^{\geq}(X), B_C^{\geq}(X)} \right\}$ *forms a nano topology [36,37] on U with respect to X. As well,* $\boldsymbol{\beta_C^{\geq}(X)} = \left\{ \boldsymbol{U, U_C^{\geq}(X), L_C^{\geq}(X)} \right\}$ *is the basis for* $\boldsymbol{\tau_C^{\geq}(X)}$*. Furthermore, CORE(C) =* $\left\{ \boldsymbol{a \epsilon C} : \boldsymbol{\beta_C^{\geq}} \neq \boldsymbol{\beta_{C-\{a\}}^{\geq}} \right\} = \cap$ *red(C) where red(C) denotes the criterion reduction.*

Definition 10. *Consider an information system S = (U, A, V, f) consisting of m entities or objects* $a_1, a_2, \ldots, a_m$*. Let the attribute set A have n members. Then, S is expressed as an* $m \times n$ *matrix with rows as objects and columns as attributes. Attributes can be designated as dimensions and each* $a_i = (a_{i1}, a_{i2}, \ldots, a_{in})$*;* $i = 1, 2, \ldots, m$ *will be a point in n-dimensional space S.*

Definition 11. *Let us consider* $a_i = (a_{i1}, a_{i2}, \ldots, a_{in})$*, being the points in n-dimensional space, then the distance* $d(a_i, C_j)$ *between* a_i*;* $i = 1, 2, \ldots, n$ *and cluster* C_j*;* $j = 1, 2, \ldots, k$ *is defined as follows.*

$$d(a_i, C_j) = \frac{||a_i - c_j||^2}{\sum_{t=1}^{k} ||a_i - c_t||^2} \tag{6}$$

where c_j *is the* C_j*,'s centroid and* $d(a_i, C_j) \in [0, 1]$*.*

Definition 12. *Support and core of a fuzzy set. The support of a fuzzy set A in X is the crisp set containing every element of X with membership grades greater than zero in A and is denoted by* $S(A) = \{x \in X;\ \mu_A(x) > 0\}$*, whereas the core of A in X is the crisp set containing every element of X with membership grades equal to 1 in A [see e.g., [59]]. Obviously, core* $[t_1, t_2] = [t_1, t_2]$*, since a closed interval* $[t_1, t_2]$ *is an equi-fuzzy interval with membership 1 [see e.g., [56–59]].*

Definition 13. *Set Superimposition. Set superimposition (S), an operation, was proposed in [56] as follows;*

$$A_1\ (S)\ A_2 = (A_1 - A_2)^{(1/2)}\ (+)\ (A_1 \cap A_2)^{(1)}\ (+)(A_2 - A_1)^{(1/2)} \tag{7}$$

where $(A_1 - A_2)^{(1/2)}$ and $(A_2 - A_1)^{(1/2)}$ are fuzzy sets [57,59] with constant membership value (1/2), and (+) signifies a union of disjoint sets. To elaborate it, let $A_1 = [s_1, t_1]$ and $A_2 = [s_2, t_2]$ are two real intervals such that when $A_1 \cap A_2 \neq \phi$, we will obtain a superimposed part. When two intervals are superimposed, each interval contributes half of its value to the superimposed interval, so from Equation (7) we obtain:

$$[s_1, t_1](S)[s_2, t_2] = [s_{(1)},t_{(2)}]^{(1/2)} (+) [s_{(2)},t_{(1)}]^{(1)} (+) (s_{(1)},t_{(2)}]^{(1/2)} \quad (8)$$

where $s_{(1)} = min(s_1, s_2)$, $s_{(2)} = max(s_1, s_2)$, $t_{(1)} = min(t_1, t_2)$, and $t_{(2)} = max(t_1, t_2)$. The superimposition process is presented using Figures 1–3 below.

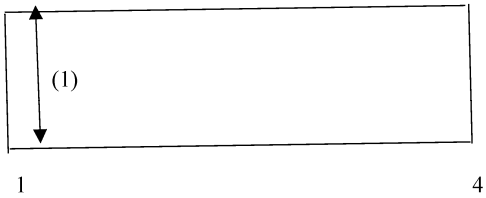

Figure 1. Interval [1, 4].

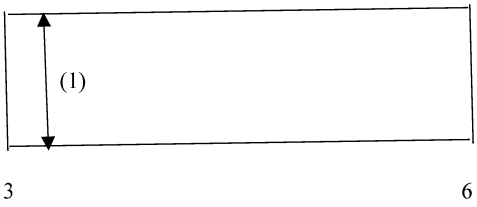

Figure 2. Interval [3, 6].

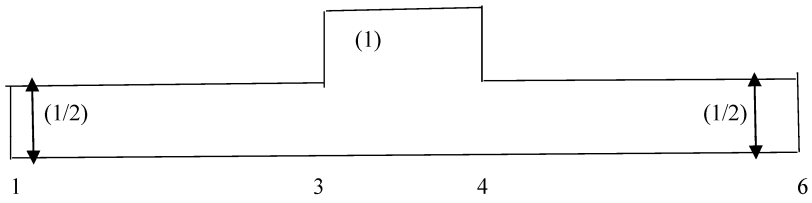

Figure 3. Superimposed interval $[1, 3]^{(1/2)} + [3, 4]^{(1)} + [4, 6]^{(1/2)}$.

Similarly, three intervals $[s_1, t_1]$, $[s_2, t_2]$, and $[s_3, t_3]$, (with non-empty intersection) are superimposed to obtain the following expression.

$$[s_1, t_1](S)[s_2, t_2](S)[s_3,t_3] = [s_{(1)},s_{(2)}]^{(1/3)} (+)[s_{(2)},s_{(3)}]^{(2/3)} (+) [s_{(3)},t_{(1)}]^{(1)} (+) [t_{(1)},t_{(2)}]^{(2/3)} (+)[t_{(2)},t_{(3)}]^{(1/3)} \quad (9)$$

where the sequence $\{s_{(i)}; i = 1, 2, 3\}$ is arranged from $\{s_i; i = 1, 2, 3\}$ in an increasing order of magnitude and $\{t_{(i)}; i = 1, 2, 3\}$ is also arranged from $\{t_i; i = 1, 2, 3\}$ in the similar fashion. Let $[s_i, t_i]$, $i = 1,2, \ldots, n$, be n real intervals with $\bigcap_{i=1}^{n} [s_i, t_i] \neq \phi$. Using generalization (9) gives as follows.

$$\begin{aligned}[s_1, t_1](S)\,[s_2, t_2](S)\,\ldots\,(S)[s_n, t_n] = {} & [s_{(1)}, s_{(2)}]^{(1/n)} (+)\,[s_{(2)}, s_{(3)}]^{(2/n)} (+)\,\ldots\,(+)\,[s_{(r)}, s_{(r+1)}]^{(r/n)} (+)\,\ldots\,(+) \\ & [s_{(n)},t_{(1)}]^{(1)}(+)[t_{(1)},t_{(2)}]^{((n-1)/n)}(+)\ldots(+)[t_{(n-r)},t_{(n-r+1)}]^{(r/n)}(+)\ldots(+)[t_{(n-2)},t_{(n-1)}]^{(2/n)}(+)[t_{(n-1)},t_{(n)}]^{(1/n)}\end{aligned} \quad (10)$$

In (10), the sequence $\{s_{(i)}\}$ is organized from $\{s_i\}$ in increasing order of magnitude for $i = 1,2, \ldots, n$ and similarly $\{t_{(i)}\}$ is also organized from $\{t_i\}$ in increasing order of magnitude [57]. It is to be noted here that the membership functions are a mixture of an empirical probability distribution function and a complementary probability distribution function given, as follows:

$$\gamma_1(x) = \begin{cases} 0, & x < s(1) \\ \frac{r-1}{m}, & s(r-1) < x < s(r) \\ 1, & x > s(m) \end{cases} \quad (11)$$

And

$$\gamma_2(x) = \begin{cases} 1, & x < t(1) \\ 1 - \frac{r-1}{n}, & t(r-1) < x < t(r) \\ 0, & x > t(n) \end{cases} \quad (12)$$

The membership function of the fuzzy interval [57,59] will be provided for us by Equations (11) and (12), which together use the Glivenko-Cantelli Lemma of order statistics [58].

Definition 14. *Match ratio. If n be the number of periods in the lifespan of a dataset (no. of years/months/days etc.) and m be the number of time-intervals in the list of lifespans of any cluster, then m/n is called the match ratio of the cluster. Obviously, $0 \leq m/n \leq 1$.*

Definition 15. *Fully/Partially periodic cluster. A cluster with a superimposed time interval is said to be full periodic if its match ratio is equal to 1. Otherwise, it is partially periodic.*

Definition 16. *Fuzzy periodic cluster. Since using the Glivenko–Calntelli lemma of order-statistics [58] on superimposed time intervals, fuzzy time intervals [57,59] can be found, the cluster associated with the superimposed time-intervals is known as a fuzzy periodic cluster.*

4. Proposed Algorithm

For detecting anomalies, a partitioning subspace clustering approach is employed. The method first uses a rough set theoretic approach for attribute or dimension reduction and then uses a dynamic *k*-means clustering approach for finding clusters along with their lifespans. Each cluster will have a sequence of time intervals representing its lifespan. Then, an interval superimposition-based approach is employed to find the periodic clusters along with the noises. The proposed method is described as follows. Here the dataset $S = (U, A)$ is an information system consisting of both conditional attributes and decision attributes. First of all, the data pre-processing techniques are employed to convert the information system into a set-valued ordered information system. Then, a dominance relation is generated on the ordered information system. With reference to the dominance relation, a nano topology and its basis is generated. Then the criterion reduction process is used to generate CORE(A) as a subset of attribute set A and the new information system $E = (U, \text{CORE}(A))$ on U is formed, which is a lower dimensional space. The pseudocode of the Algorithm 1 for the criterion reduction is given below.

Algorithm 1: Subspace Generation

Input: (U, A): the information system, where the attribute set A is divided into C-conditional attributes and D-decision attributes, consisting of n objects,
Output: Subspace of (U, A)
Step 1. Generate a dominance relation $R_C^{\geq}$ on U corresponding to C and $X \subseteq U$.
Step 2. Generate the nano topology $\tau_C^{\geq}(X)$ and its basis $\beta_C^{\geq}(X)$
Step 3. **for** each $x \in C$, find $\tau_{C-\{x\}}^{\geq}(X)$ and $\beta_{C-\{x\}}^{\geq}(X)$
Step 4. **if** ($\beta_C^{\geq}(X) = \beta_{C-\{x\}}^{\geq}(X)$)
Step 5. **then** drop x from C,
Step 6. **else** form criterion reduction
Step 7. **end for**
Step 8. generate CORE(C) = $\cap$ {criterion reductions}
Step 9. Generate subspace of the given information system.

The above algorithm supplies the CORE of the attribute set by removing insignificant attributes which gives us a subspace $E = (U, \text{CORE}(A))$ of the given information system $S = (U, A)$. Then a dynamic *k*-means is applied to E. The following is an explanation of the algorithm: First of all, it randomly picks first *k*–data instances from the CORE(A) as *k*-clusters-centroids with associated timestamps (times of generation) as the *start-time*

of their lifespans. For each cluster, a *last-time* and a list are maintained to keep the last timestamp and lifespan of each cluster, respectively. Initially *start-time* = *last-time*. If a data instance is added to a cluster based on how far away from the cluster centroid it is, its current time-stamp (*current-time*) is added to the lifespan to obtain an updated life span, provided that the time gap between the cluster's *last-time* and the data instance *current-time* is within a predetermined range, such as $t_{\max}$. Otherwise, a new life-span will start by setting the *current-time* as a *start-time* and the previous life-span of the cluster will be closed with *last-time* as the end of the life-span. The lifespan of the cluster will be put on the list maintained for it if its length is greater than a specified length (say $t_{\min}$). The lifespans of the earlier and later clusters are updated if a data instance switches from one cluster to another during the execution process. For instance, if the time stamp on the outgoing data instance is either the *start-time* or *end-time* of the preceding cluster, the lifespan of the prior cluster is updated by using the next or previous cluster time-stamps respectively. Updates are made to the cluster-centroids as well. Again, the lifespans of the former and later clusters will not change if the time stamp of the outgoing data instance falls within those lifespans, but the cluster centroids will be modified. Similar to this, if the time stamp of a data instance migrating from one cluster to another falls outside the later cluster's lifespan, the cluster-centroid is updated and the later cluster's life-span is updated as well, provided that the time gap between the two clusters is within a certain limit ($t_{\max)}$. The pseudocode of the algorithm is given below.

Here each output cluster in the final output cluster set has a sequence of time intervals describing its lifespan. It should be noted that only clusters with lifespans of at least $t_{\min}$ are provided by Algorithm 2.

Algorithm 2: Dynamic *k*-means clustering algorithm

Input: *E*: Information system consisting *n* objects and attribute set CORE(A) $\subseteq A$, t_{max}: the maximum time-gap of consecutive time-stamp, t_{min}: the minimum length of lifespan.
Output: Set of clusters where each cluster is associated with a sequence of time intervals as its lifespans

Step 1. Given d_1-dimensional dataset CORE(A)
Step 2. Select $C[i] = \{x[i], \text{tp}[i]\}$; $i = 1, 2, \ldots, k$, where $x[i]$ be the data instances or means of clusters, tp[i] points to list of time-intervals each maintained for every cluster contains time-stamps (*start-time*) of $x[i]$ and *start-time* = *last-time* initially
Step 3. **for** each incoming data instance x with current time-stamp *current-time*
Step 3. {**if** $\text{d}(\text{x}, C_\text{j}) \leq \text{d}(\text{x}, C_\text{i})$, $i \neq j$; $i = 1, 2, \ldots, k$
Step 4. {Add x to C_j
Step 5. Update mean(C_j)
Step 6. **if** ($|$ *current-time* $-$ *last-time*[j] $| \leq t_{max}$)
Step 7. {**if**(*last-time*[j] $\leq$ *current-time*)
Step 8. extend lifespan(C_j) by setting *last-time*[j] = *current-time*
Step 9. **else go** to Step3
Step 10. }
Step 11. **else if** $|$ *last-time*[j] $-$ *start-time* $| \geq \text{t}_{\min}$
Step 12. {Add [*start-time*[j], last-time[j]] to tp[j]
Step 13. set *last-time*[j] = *start-time*[j] = *current-time*
Step 14. }
Step 15. }
Step 16. }
Step 17. **if** (assign does not occur) go to step19
Step 18. **else go** to Step3
Step 19. Output cluster set

For each cluster with a sufficient number of time intervals as its lifespans, the following procedure is applied to find periodic clusters from the interval list. The interval superimposition operation is to keep the information about the periods (time interval associated with a cluster). The interval superimposition is used only if the intervals have overlapping

or non-empty intersections. Throughout Algorithm 3 execution, a list of superimposed time intervals is maintained. The total number of time intervals of any clusters is taken as n (number of Years/Months etc.). To determine whether a new crisp time-interval can be superimposed on an already superimposed time-interval or not. it is checked whether the interval has a non-empty intersection with the core of the superimposed time interval or not (the definition of core is given in Section 3). If it has, then the superimposition process is computed to get a new superimposed time interval and membership values are reconstructed accordingly. The list of superimposed time intervals is initially empty. A full pass through the time interval list of a cluster is conducted during Algorithm 3 execution. When it switches to a new time interval, it determines if it can be superimposed on any of the previously obtained superimposed intervals. If so, the superimposition process is performed, which updates the relevant superimposed time interval. This time interval is added as a new entry to the list if it is not superimposed with any of the previously acquired superimposed time intervals (kept as a list). Finally, each superimposed time interval is examined to determine the number of time intervals superimposed in one place and kept using a counter (m). At the beginning of the superimposition process of a time interval, the value of m is taken as 1. If a time interval is superimposed on the time interval, then m is updated by adding 1 to it. After the execution, the match ratio for a cluster is obtained with the help of m and n. If the match ratio is found to be 1, the corresponding cluster is fully periodic, else partially periodic. Each superimposed time interval produces a fuzzy time interval. This way, the fuzzy periodic clusters can be obtained. The pseudocode for the process is given below.

The function compsuperimp(lt, lct) initially finds the intersection between lt and the core of lct. If it is found to be non-empty, the function computes the superimposition process by reconstructing the membership values. If lt has been superimposed on lct it returns 1, otherwise it returns 0. get () and append () are functions operating on time interval lists to obtain a pointer to the next time interval in a list and to append a time interval, into a list, respectively. For each cluster, a counter (m) is also kept in order to keep track of how many time intervals are superimposed in one place. The match ratio is computed with the help of m. If the match ratio is found to be 1, the corresponding cluster is fully periodic, else partially periodic. Finally, the fuzzy intervals can be generated with the help of superimposed intervals to get fuzzy periodic patterns. The flowchart for the proposed method is described in Figure 4 below.

Algorithm 3: Algorithm for finding periodic (fully/partially) and fuzzy periodic clusters

Input: Set of clusters along with their lifespans (set of sequence of time intervals).
Output: Set of fuzzy periodic clusters

```
Step 1. For each cluster c with list of linespans L.
Step 2. initially Lc=null//Lc is the list of superimposed intervals
Step 3. lt = L.get() //lt points to the 1st time interval (lifespan) in L
Step 4. Lc = append(lt)
Step 5. m = 1 //m = number of intervals superimposed
Step 6. while((lt=L.get())!=null)
Step 7.       {flag = 0
Step 8.        while ((lct =L.get())!=null)
Step 9.        if (compsuperimp(lt, lct)
Step 10.        flag =1
Step 11. if (flag == 0)
Step 12. Lc.append(lt) }
Step 13. }
Step 14.    }
Step 15. compsupeimp(lt, lct)
Step 16.    if(|intersect(lct, lt)!=null)|
```

Step 17. { superimp(*lct*, *lt*)
Step 18. *m*++
Step 19. return 1
Step 20. }
Step 21. return 0
Step 22. Compute match ratio = m/n //n = number periods in the whole dataset.
Step 23. if (match = 1)
Step 24. the cluster *c* is fully periodic
Step 25. **else** partially periodic
Step 26. generate fuzzy time intervals from superimposed time intervals to get fuzzy periodic clusters.
Step 27. **End**

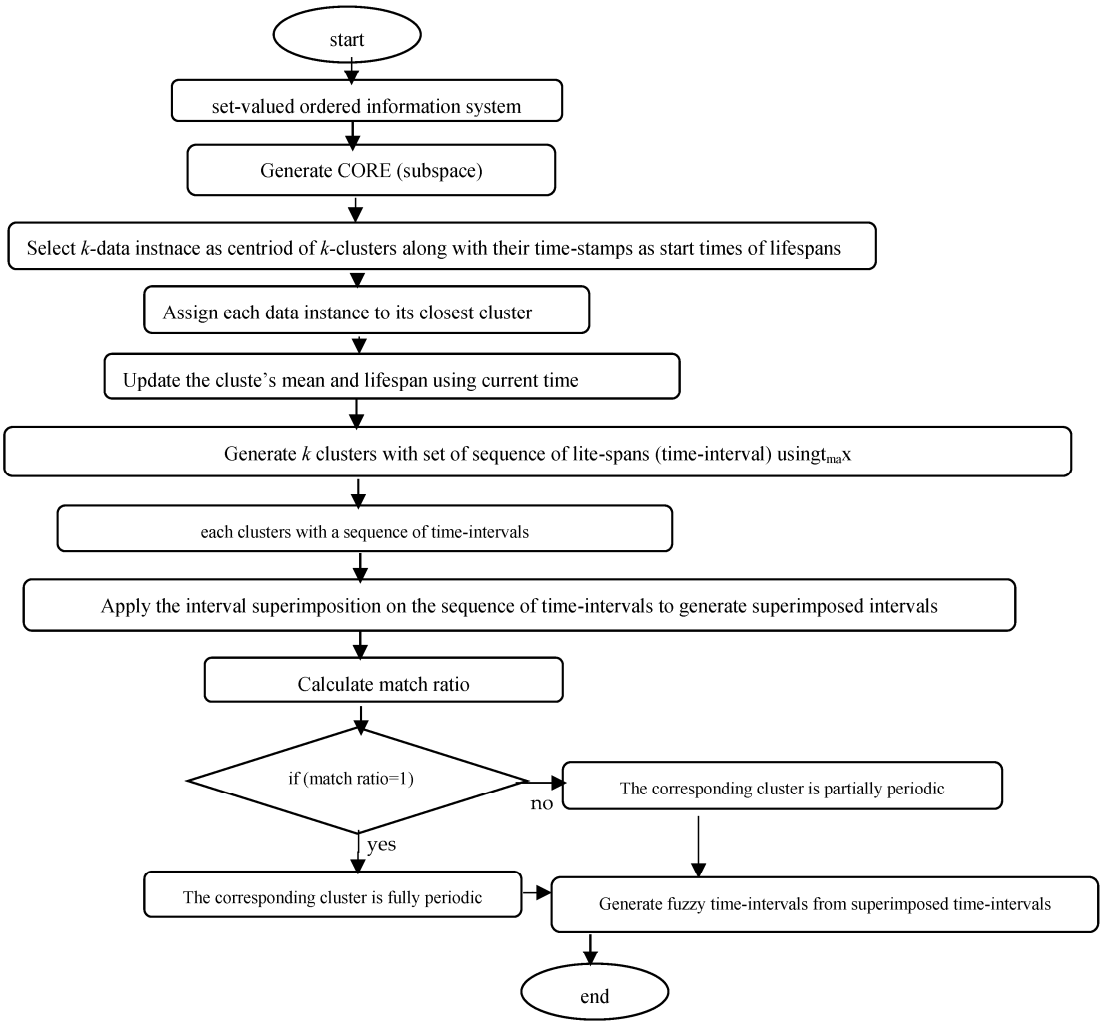

Figure 4. Flowchart of the proposed method (RADSPCA).

Anomalies are data instances or groups of data instances that either belong to sparse clusters or don't fit the defined lifespans. As a result, a data instance may be anomalous depending on both its generation time and its distance from clusters.

5. Complexity Analysis

For generating dominance classes and corresponding classes, the algorithm needs to compare the values of all the possible pairs of objects from U in all dimensions, there can be at most $|U| \times |U| \times |C|$ number of comparisons. So, the computational complexity for step1 is $O(n^2.d)$, where $|U| = n$, and $|C| = d$. For generating the nano topology,

the lower approximation and approximation of the set has to be generated, which takes computational time O(|X|.|U|). So the total computational cost of step1 and step2 is O(n^2.d+|X|.|U|) = O(n^2.d) which is the worst-case complexity. From step3 for loop starts it runs over at most all the attributes of the attribute set. The computation from step 4 to step7 takes constant time, say O(k_1), where k_1 = constant. Therefore, the computational cost from step3 to step8 is O(k_1d). Similarly, that of step9 and 10 is also constant, say O(k_2), where k_2 = constant. The overall complexity of algorithm1 is O(n^2.d + k_1d+ k_2) = O(n^2.d). For finding the complexity of Algorithm2, the following steps are taken. Let $k(\leq n)$ be the number of clusters. The computational cost of a centroid is O(n + n.k.d_1) = O(n.k.d_1), where d_1 ($\leq d$), is the dimension of the CORE. Also, O(2n.k) = O(n.k) is the time required compute the minimum distance and time-gap for each cluster. The cost of updating cluster-mean and lifespan is O(2k). The total cost of algorithm2 is O(i(n.k.d_1 + n.k + k)) = O(i.n.k.d_1) = O(n^3) as $i(\leq n)$, the number of iterations, $k \leq n$, and d_1 is considerably small. The worst-case complexity of the whole method is O(n^2.d + n^3). For finding the time-complexity of Algorithm 3, we proceed as follows. Let n_1 be the size of the sequence of time intervals associated with a cluster and n_2 be the average number of time intervals superimposed. For each time interval of a cluster, it is required to make a pass through the list of superimposed time intervals to check whether the corresponding time interval can be superimposed on any of the available superimposed time intervals or not. For this, the intersection of the current time-interval with the core of the superimposed time interval is computed, which requires O(1) time. If the current time interval is superimposed, then its boundaries have to be inserted into two sorted arrays used to keep the end points of the superimposed time-intervals (one sorted array for left end points and other for right end points). Now, searching in a sorted array requires O(log n_1) time and insertion needs O(n_1) time. The two end points require O(2(log n_1 + n_1)) = O(n_1) time. For one cluster, the process requires O(n_1.p.n_2) time, where p is the size of the list of superimposed time intervals. On the other hand, p = O(n_1), and n_2 = O(n_1); the overall time-complexity in the worst-case is O($n_1{}^3$). For k clusters, the total time-complexity in worst-case is O(k.$n_1{}^3$). Therefore, the worst-case complexity of the whole method is O((n^2.d + n^3) + k.$n_1{}^3$). Also k = O(n), which gives the time-complexity, as O(n^2.d + n^3 + n.$n_1{}^3$) = O(n^3 + n.$n_1{}^3$), as $d \leq n$, which is the time-complexity of the method in worst-case. Since the time-complexity of the method depends on n and n_1, and not on d (dimension), the worst-case complexity of the method can be rewritten as O(n^3). Thus, the method runs in cubic time.

6. Experimental Analysis and Results

In this Section the experimental studies are conducted and comparative analysis of the proposed method is performed against ten different clustering-based anomaly detection algorithms [see e.g., Table 3], namely k-means [19], IF (Isolation Forest) [61,62], SC (Spectral Clustering) [54], HDBSCAN (hierarchical density-based spatial clustering of applications with noise) [63], ACA (Agglomerative Clustering Algorithm) [54], LOF (Local Outlier Factor) [54], SSWLOFCC (streaming sliding window local outlier factor coreset clustering algorithm) [54], PCM (Partitioning Clustering with Merging) [10], OnCAD (Online Clustering and Anomaly Detection) [46], and MCA (Mixed Clustering Algorithm) [9]. The dataset employed for the experiment is Kitsune Network Attack dataset [64] and KDDCUP'99 dataset [64], collected through the UCI machine repository. The Kitsune Network attack dataset [65] is a multi-variate, sequential, time-series dataset with real and temporal attributes. It has 27,170,754 data instances and its number of attributes is 115. It is a collection of nine network attack datasets each containing network packets and various cyberattacks, collected from an IoT-based network system or commercial IP-based surveillance system. The dataset, KDDCUP'99 [64] is a multi-variate dataset with numeric, categorical and temporal attributes. It has 4,898,431 data instances with 37 numeric, 3 categorical and 1 temporal (time-stamp) attributes.

The proposed method (RADSPCA) is first implemented with the KDDCUP'99 [64] dataset, using MATLAB. The implementation process consists of three stages: the input

data pre-processing, periodic subspace clustering, and testing. First of all, the method accepts the input data and converts it to a set-valued matrix. The matrix representation of the dataset is the information system. Since the rough set can't deal with continuous attributes, so they are discretized at the same time. The Algorithm 1 is then applied to find the subset of the attribute set by removing the insignificant attributes and by using the concept of dominance relation, nano topology and its basis. The algorithm1 gives the subset as CORE of the attribute set. Then Algorithm 2 is applied on the CORE to find clusters along with the set of sequence time-intervals where each cluster is associated with a sequence of time intervals describing its lifespan. For the efficient implementation, two parameters, namely t_{min} (minimum length of a lifespan = 180 min) and t_{max} (maximum time-gap between two consecutive time-stamps associated with a cluster = 20 min) are to be specified. Then the Algorithm 3 is applied to the clusters to generate periodic, partially periodic, and fuzzy periodic clusters. The performances of the proposed method along with the afore-mentioned methods are recorded. The performance is measured using the following evaluation metrics.

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$\text{F1} - \text{score} = \frac{2 * (Precision * Recall)}{Precision + Recall}$$

The details of the outcomes of the investigations are presented in tabular form in Table 1 below.

Table 1. Comparative performances analysis of RADSPCA with some well-known existing methods using KDDCUP'99 [64] dataset.

	Algorithms	**Evaluation Metrics**			**Execution Time (in Seconds)**	**Periodic Clusters Obtained**
		Recall	**Precision**	**F1-Score**		
1	*k*-means	0.9605	0.9400	0.9500	28	×
2	IF model	0.8301	0.850	0.8400	19	×
3	SC	0.6220	0.6004	0.6110	44	×
4	HDBSCAN	0.2530	0.2300	0.2410	95	×
5	ACA	0.8400	0.8010	0.8200	16	×
6	LOF	0.9550	0.9390	0.9470	14	×
7	SSWLOFCC	0.9665	0.9460	0.9560	12	×
8	PCM	0.8800	0.8420	0.8600	26	×
9	OnCAD	0.9751	0.9650	0.9700	30	×
10	MICA	0.9822	0.9780	0.9800	28	×
11	Proposed Approach (RADSPCA)	0.9812	0.9790	0.9800	58	√

Similarly, the proposed method (RADSPCA) is also implemented with the Kitsune Network attack dataset [64] and the results were recorded in tabular form in Table 2 below.

Table 2. Comparative performances analysis of RADSPCA with some well-known existing methods using Kitsune [65] dataset.

	Algorithms	Evaluation Metrics			Execution Time (in Seconds)	Periodic Clusters Obtained
		Recall	Precision	F1-Score		
1	*k*-means	0.8701	0.8501	0.8600	95	×
2	IF model	0.7300	0.7502	0.7400	64.5	×
3	SC	0.6645	0.6420	0.6530	149.5	×
4	HDBSCAN	0.3899	0.3793	0.3850	150	×
5	ACA	0.7410	0.7010	0.7200	54.4	×
6	LOF	0.90401	0.9000	0.9020	47.6	×
7	SSWLOFCC	0.9280	0.9499	0.9390	40	×
8	PCM	0.7430	0.7810	0.7600	88	×
9	OnCAD	0.8450	0.8353	0.8400	102	×
10	MICA	0.9832	0.9770	0.9800	68	×
11	Proposed Approach (RADSPCA)	0.9860	0.9801	0.9830	88.5	√

The following observations can be drawn from the obtained results.

The *k*-means algorithm is quite good as per as the values of the evaluation metrics are concerned. However, it is sensitive to both the dataset and the dimensions. It is also sensitive to the distribution of the dataset in the plane. It cannot supply periodic clusters.

The IF model is reasonably good; however, it is efficient up to a certain dimensional dataset, beyond which its efficacy decreases rapidly. It cannot supply periodic clusters.

The SC and HDBSCAN algorithms are poor in both performance and execution times. Though. HDBSCAN works very well with lower-dimensional data, but its performance decreases proportionately with the increase in the dimension of the dataset. Both algorithms are not capable of finding periodic clusters.

The ACA is reasonably good as far as performance and execution time are concerned. However, it is very sensitive to the order of input to the algorithm. It is not useful for determining periodic clusters.

Though the LOF algorithm performs well, it has similar issues as the *k*-means algorithm.

The SSWLOFCC performs better, and its performance does not depend much on the size of the dataset. However, its execution time increases with the increment of the dimension as well as the size of the dataset. It cannot extract periodic clusters.

The PCM is an algorithm consisting of both k-means and hierarchical agglomerative approaches. However, its performance decreases with the increase in size and dimension of the dataset. It cannot be used for finding periodic clusters.

OnCAD has a problem with dimensionality. Its accuracy and execution time fall rapidly with the increase in data size and dimensions. It cannot find periodic clusters.

The MICA is very good as far as accuracy is concerned. It has recalls of 0.9822 and 0.9832, precisions of 0.978 and 0.977, and F1-scores of 98% and 98% with the KDDCUP'99 [65] and the Kitsune [64] datasets, respectively, which is quite impressive. Its execution time is also quite good. However, it cannot be used for finding periodic clusters.

The proposed algorithm (RADSPCA) is better as far as performance is concerned. It has recall, precision, and F1-score values that are almost the same for both datasets. It has recall 0.9812, 0.9860, precision 0.979, 0.9801, and F1-score 0.98, 0.983, with the datasets KDDCUP'99 [64] and Kitsune [65], respectively. It has the ability to extract periodic clusters, which others do not. Though its execution time is a little longer than that of others, the rate of increase is quite low. The extra is the time spent finding subspace and extracting

periodicity. Thus, the execution time of RADSPCA depends mostly on the dataset sizes and the number of periods associated with a cluster in its lifespan.

Moreover, the RADSPCA's execution time in seconds is compared against that of *k*-means [19], IF model [61,62], SC Algorithm [54], HDBSCAN algorithm [63], ACA Algorithm [54], LOF algorithm [54], SSWLOFCC algorithm [54], PCM algorithm [10], OnCAD algorithm [46], and MCA Algorithm [9] and the results are presented using a bar diagram in Figure 5.

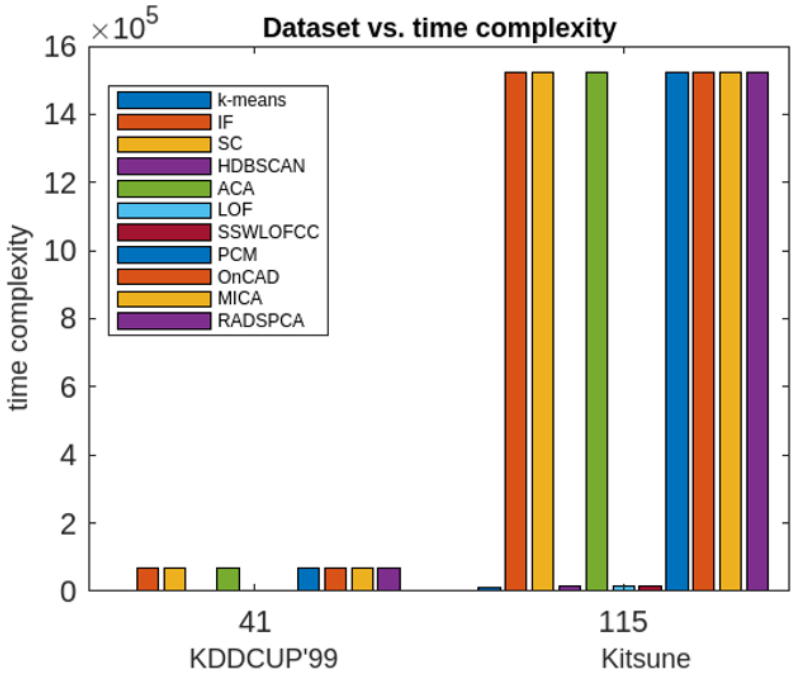

Figure 5. Comparative analysis of all the aforesaid methods in terms of time complexity.

It has been found from Figure 5 that most of the aforesaid algorithms depend not only on dataset sizes but also on dimensions, and some are inefficient with high-dimensional data. However, RADSPCA is less dependent on the dimension of the dataset and is a bit dependent on the sequence of time intervals associated with every cluster, which is a negligible factor. In fact, Figures 6 and 7 further validate that the RADSPCA's execution time grows linearly with respect to the increase in dimension if the dataset size is kept constant. Also, RADSPCA runs in cubic time.

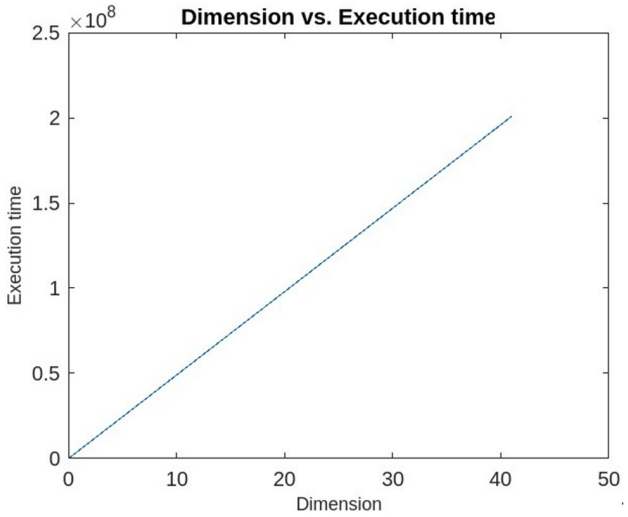

Figure 6. The execution time with respect to the dimension using KDDCUP'99 dataset [64].

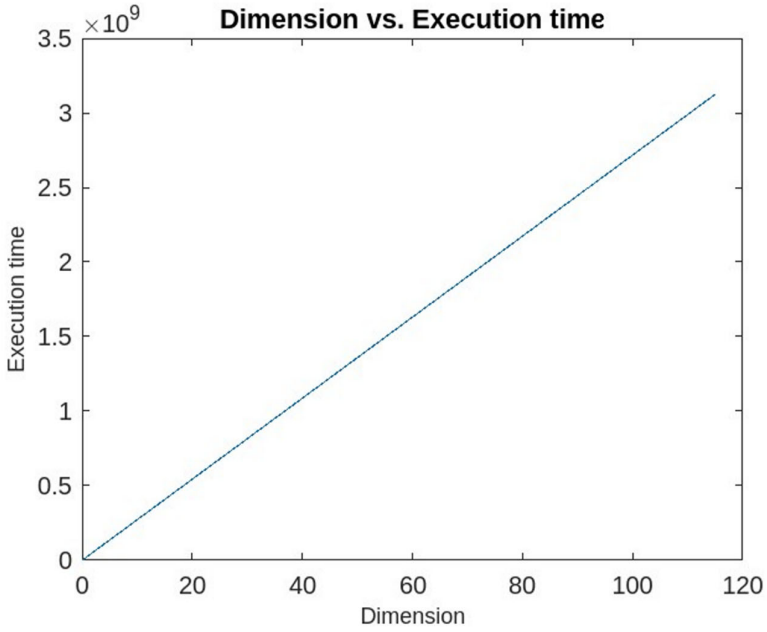

Figure 7. The execution time with respect to the dimension using Kitsune dataset [65].

7. Conclusions, Limitations and Lines for Future Works

7.1. Conclusions

In this article, a clustering-based method for finding real-time anomalies in a subspace is given. The method first uses a nano-topology-based attribute reduction approach for finding subspace as the core of an attribute set. Then, a dynamic k-means clustering approach is employed to find k-clusters in the subspace. It is to be mentioned here that the clusters obtained by the aforesaid approach will have a k-number of sequences of time intervals, and each cluster will be associated with a sequence of time intervals describing its lifespan. Then, using an interval superimposition method, a superimposed time interval is obtained, and a match ratio for each cluster is also computed. The match ratio determines whether the cluster is fully or partially periodic. Further, from each superimposed time interval, a fuzzy time interval can be computed, and the cluster associated with the fuzzy time interval is termed a fuzzy periodic cluster. At the end, the method supplies fuzzy periodic clusters in the subspace. Since the obtained clusters are periodic in nature, they provide more detailed information about the nature of the data instances. The anomalies would be mostly doubtful instances that either belong to sparse clusters or do not belong to any of the periodic clusters.

The time-complexity of the method is computed and found to be $O(n^3 + n.{n_1}^3)$ in the worst-case, where n = the number of instances and n_1 = the maximum number of intervals associated with any cluster. Obviously, n_1 is very small in comparison to n. Therefore, the method runs in cubic time. Further, it has also been found that RADSPCA runs linearly with respect to the dimension of the datasets.

To find efficacy further, ten well-known clustering-based algorithms were taken, and a detailed comparative analysis was conducted against RADSPCA, first using the KDDCUP'99 [64] dataset and then the Kitsune [65] network attack dataset. Experimentally, RADSPCA is found to be more efficient than others in terms of recall, precision, and F1-score in high-dimensional data.

7.2. Limitations and Future Directions of Work

The proposed RADSPCA has some limitations. Firstly, it is unable to deal with continuous data, as a rough set is inefficient to handle continuous data. Secondly, the method uses the k-means algorithm. It has the following issues with finding anomalies: For example, the centroid of any cluster can be pulled by anomalies, or there may be a

cluster of anomalies extracted by the method that looks such as a normal cluster. Finally, the method cannot detect anomalies from temporal interval data.

Future works could be possible in the following lines.

Methods other than the *k*-means approach can be employed for efficient anomaly detection.

An effective method can be proposed to deal with continuous attributes or temporal interval datasets. Table 3 shows acronyms and their full form and purpose.

Table 3. Acronym table.

Acronym	Full Form and Purpose
IF	Isolation Forest: It is an anomaly detection using binary tree.
SC	Spectral Clustering: It has been used as an outlier detection algorithm many times
HDBSCAN	Hierarchical Density-based Spatial Clustering of Applications with Noise: It is a density–based hierarchical clustering approach that has been used for anomaly detection many times with less efficacies
ACA	Agglomerative Clustering Algorithm: It is a hierarchical clustering approach for anomaly detection.
LOF	Local Outlier Factor: It is an algorithm to identify outliers based on local neighborhood.
SSWLOFCC	Streaming Sliding Window Local Outlier Factor Coreset Clustering Algorithm: It focuses on real-time detection of anomalies using big data technologies.
PCM	Partitioning Clustering with Merging: It is an algorithm for finding anomalies which uses both partitioning and Hierarchical approaches
OnCAD	Online Clustering and Anomaly Detection: It is a clustering-based anomaly detection approach in data streams that considers the temporal as well as spatial proximity of observations to detect the real-time anomaly.
MICA	Mixed Clustering Algorithm: It is an algorithm for finding real-time anomalies using both partitioning and Hierarchical approaches
RADSPSCA	Real-time Anomaly Detection with Subspace Periodic Clustering Approach is the method proposed in this article.

Author Contributions: Conceptualization, F.A.M. and M.S.; Methodology, F.A.M. and M.S.; Software, F.A.M. and M.S.; Validation, F.A.M. and M.S.; Formal Analysis, F.A.M.; Investigation, F.A.M. and M.S.; Resource, F.A.M. and M.S.; Data Curation, F.A.M. and M.S.; Writing—original draft preparation, F.A.M.; writing—review and editing, F.A.M. and M.S.; visualization, F.A.M. and M.S.; supervision, F.A.M.; project administration, M.S. and F.A.M.; funding acquisition, M.S. All authors have read and agreed to the published version of the manuscript.

Funding: The corresponding author states that the work does not have any external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data, code and other materials can be made available on request.

Conflicts of Interest: There is no conflict of interest or competing interests among the authors.

References

1. Xu, L.D.; He, W.; Li, S. Internet of Things in Industries: A Survey. *IEEE Trans. Ind. Inform.* **2014**, *10*, 2233–2243. [CrossRef]
2. Sisinni, E.; Saifullah, A.; Han, S.; Jennehag, U.; Gidlund, M. Industrial Internet of Things: Challenges, Opportunities, and Directions. *IEEE Trans. Ind. Inform.* **2018**, *14*, 4724–4734. [CrossRef]
3. Sethi, P.; Sarangi, S. Internet of Things: Architectures, Protocols, and Applications. *J. Electr. Comput. Eng.* **2017**, *2017*, 9324035. [CrossRef]
4. Papaioannou, M.; Karageorgou, M.; Mantas, G.; Sucasas, V.; Essop, I.; Rodriguez, J.; Lymberpoulos, D. A Survey on Security Threats and Countermeasures in Internet of Medical Things (IoMT). *Trans. Emerg. Telecommun. Technol.* **2020**, *33*, e4049. [CrossRef]
5. Mantas, G.; Komninos, N.; Rodriguz, J.; Logota, E.; Marques, H. Security for 5G Communications. In *Fundamentals of 5G Mobile Networks*; Wiley: Hoboken, NJ, USA, 2015; pp. 207–220. [CrossRef]
6. Zarpelão, B.B.; Miani, R.S.; Kawakami, C.T.; de Alvarenga, S.C. A survey of intrusion detection in Internet of Things. *J. Netw. Comput. Appl.* **2017**, *84*, 25–37. [CrossRef]
7. Makhdoom, I.; Abolhasn, M.; Lipman, J.; Liu, R.P.; Ni, W. Anatomy of Threats to the Internet of Things. *IEEE Commun. Surv. Tutorials* **2019**, *21*, 1636–1675. [CrossRef]
8. Zachos, G.; Essop, I.; Mantas, G.; Porfyrkis, K.; Ribeiro, J.C.; Rodriguez, J. Generating IoT Edge Network Datasets based on the TON_IoT Telemetry Dataset. In Proceedings of the IEEE 26th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD-2021), Porto, Portugal, 25–27 October 2021. [CrossRef]
9. Mazarbhuiya, F.A.; Shenify, M. A Mixed Clustering Approach for Real-Time Anomaly Detection. *Appl. Sci.* **2023**, *13*, 4151. [CrossRef]
10. Mazarbhuiya, F.A.; AlZahrani, M.Y.; Mahanta, A.K. Detecting Anomaly Using Partitioning Clustering with Merging. *ICIC Express Lett.* **2020**, *14*, 951–960.
11. Mazarbhuya, F.A.; AlZahrani, M.Y.; Georgieva, L. *Anomaly Detection Using Agglomerative Hierarchical Clustering Algorithm*; ICISA 2018. Lecture Notes on Electrical Engineering (LNEE); Springer: Hong Kong, China, 2019; Volume 514, pp. 475–484.
12. Mazarbhuiya, F.A. Detecting Anomaly using Neighborhood Rough Set based Classification Approach. *ICIC Express Lett.* **2023**, *17*, 73–80.
13. Al Mamun, S.M.A.; Valmaki, J. Anomaly Detection and Classification in Cellular Networks Using Automatic Labeling Technique for Applying Supervised Learning. *Procedia Comput. Sci.* **2018**, *140*, 186–195. [CrossRef]
14. Liu, Y.; Wang, H.; Zhang, X.; Tian, L. An Efficient Framework for Unsupervised Anomaly Detection over Edge-Assisted Internet of Things. *ACM Trans. Sens. Netw.* **2023**, *2023*, 1–26. [CrossRef]
15. Mozaffari, M.; Doshi, K.; Yilmaz, Y. Self-Supervised Learning for Online Anomaly Detection in High-Dimensional Data Streams. *Electronics* **2023**, *12*, 1971. [CrossRef]
16. Angiulli, F.; Fasetti, F.; Serrao, C. Anomaly detection with correlation laws. *Data Knowl. Eng.* **2023**, *145*, 102181. [CrossRef]
17. Fan, Z.; Wang, G.; Zhang, K.; Liu, S.; Zhong, T. Semi-Supervised Anomaly Detection via Neural Process. *IEEE Trans. Knowl. Data Eng.* **2023**, *2023*, 1–13. [CrossRef]
18. Lu, T.; Wang, L.; Zhao, X. Review of Anomaly Detection Algorithms for Data Streams. *Appl. Sci.* **2023**, *13*, 6353. [CrossRef]
19. Hartigan, J.A. *Hartigan Clustering Algorithms*; John Wiley & Sons: Hoboken, NJ, USA, 1975.
20. Cheng, Y.-M.; Jia, H. A Unified Metric for Categorical and Numeric Attributes in Data Clustering. Hong Kong University Technical Report. 2011. Available online: https://www.comp.hkbu.edu.hk/tech-report (accessed on 12 June 2018).
21. Mazarbhuiya, F.A.; Abulaish, M. Clustering Periodic Patterns using Fuzzy Statistical Parameters. *Int. J. Innov. Comput. Inf. Control.* **2012**, *8*, 2113–2124.
22. Gil-Garcia, R.; Badia-Contealles, J.M.; Pons-Porrata, A. Dynamic Hierarchical Compact Clustering Algorithm. In *Progress in Pattern Recognition, Image Analysis and Applications*; Sanfeliu, A., Cortés, M.L., Eds.; CIARP 2005, LNCS 3775; Springer: Berlin/Heidelberg, Germany, 2005; pp. 302–310.
23. Hammouda, K.M.; Kamel, M.S. Efficient phrase-based document indexing for Web document clustering. *IEEE Trans. Knowl. Data Eng.* **2004**, *16*, 1279–1296. [CrossRef]
24. Erfani, S.M.; Rajasegrar, S.; Karunasekera, S.; Leckie, C. High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning. *Pattern Recognit.* **2016**, *58*, 121–134. [CrossRef]
25. Hodge, V.; Austin, J. A survey of outlier detection methodologies. *Artif. Intell. Rev.* **2004**, *22*, 85–126. [CrossRef]
26. Kaya, M.; Schoop, M. Analytical Comparison of Clustering Techniques for the Recognition of Communication Patterns. *Group Decis. Negot.* **2022**, *31*, 555–589. [CrossRef]
27. Aggarwaal, C.C.; Philip, S.Y. An effective and efficient algorithm for high-dimensional outlier detection. *VLDB J.* **2005**, *14*, 211–221. [CrossRef]
28. Ramchandran, A.; Sangaiaah, A.K. Chapter 11—Unsupervised Anomaly Detection for High Dimensional Data—An Exploratory Analysis. In *Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications*; Intelligent Data-Centric Systems; Academic Press: Cambridge, MA, USA, 2018; pp. 233–251.
29. Retting, L.; Khayati, M.; Cudre-Maurooux, P.; Piorkowski, M. Online anomaly detection over Big Data streams. In Proceedings of the 2015 IEEE International Conference on Big Data, Santa Clara, CA, USA, 29 October–1 November 2015.
30. Alguliyev, R.; Aliguuliyev, R.; Sukhostat, L. Anomaly Detection in Big Data based on Clustering. *Stat. Optim. Inf. Comput.* **2017**, *5*, 325–340. [CrossRef]

31. Hahsler, M.; Piekenbroock, M.; Doran, D. dbscan: Fast Density-Based Clustering with *R*. *J. Stat. Softw.* **2019**, *91*, 1–30. [CrossRef]
32. Song, H.; Jiang, Z.; Men, A.; Yang, B. A Hybrid Semi-Supervised Anomaly Detection Model for High Dimensional Data. *Comput. Intell. Neurosci.* **2017**, *2017*, 8501683. [CrossRef] [PubMed]
33. Mazarbhuiya, F.A. Detecting IoT Anomaly Using Rough Set and Density Based Subspace Clustering. *ICIC Express Lett.* **2022**. *accepted*. [CrossRef]
34. Ahmed, S.; Lavin, A.; Purdy, S.; Aghaa, Z. Unsupervised real-time anomaly detection for streaming data. *Neurocomputing* **2017**, *262*, 134–147. [CrossRef]
35. Pawlak, Z. Rough sets. *Int. J. Comput. Inf. Sci.* **1982**, *11*, 341–356. [CrossRef]
36. Thivagar, M.L.; Richaard, C. On nano forms of weakly open sets. *Int. J. Math. Stat. Invent.* **2013**, *1*, 31–37.
37. Thivagar, M.L.; Priyalaatha, S.P.R. Medical diagnosis in an indiscernibility matrix based on nano topology. *Cogent Math.* **2017**, *4*, 1330180. [CrossRef]
38. Kim, B.; Alawaami, M.A.; Kim, E.; Oh, S.; Park, J.; Kim, H. A Comparative Study of Time Series Anomaly Detection, Models for Industrial Control Systems. *Sensors* **2023**, *23*, 1310. [CrossRef]
39. Alghawli, A.S. Complex methods detect anomalies in real time based on time series analysis. *Alex. Eng. J.* **2022**, *61*, 549–561. [CrossRef]
40. Younas, M.Z. Anomaly Detection using Data Mining Techniques: A Review. *Int. J. Res. Appl. Sci. Eng. Technol.* **2020**, *8*, 568–574. [CrossRef]
41. Thudumu, S.; Branch, P.; Jin, J.; Siingh, J. A comprehensive survey of anomaly detection techniques for high dimensional big data. *J. Big Data* **2020**, *7*, 42. [CrossRef]
42. Habeeb, R.A.A.; Nasaaruddin, F.; Gani, A.; Hashem, I.A.T.; Ahmed, E.; Imran, M. Real-time big data processing for anomaly detection: A Survey. *Int. J. Inf. Manag.* **2019**, *45*, 289–307. [CrossRef]
43. Wang, B.; Hua, Q.; Zhang, H.; Tan, X.; Nan, Y.; Chen, R.; Shu, X. Research on anomaly detection and real-time reliability evaluation with the log of cloud platform. *Alex. Eng. J.* **2022**, *61*, 7183–7193. [CrossRef]
44. Halstead, B.; Koh, Y.S.; Riddle, P.; Pechenizkiy, M.; Bifet, A. Combining Diverse Meta-Features to Accurately Identify Recurring Concept Drift in Data Streams. *ACM Trans. Knowl. Discov. Data* **2023**, *17*, 1–36. [CrossRef]
45. Zhao, Z.; Birke, R.; Han, R.; Robu, B.; Buchenak, S.; Ben Mokhtar, S.; Chen, L.Y. RAD: On-line Anomaly Detection for Highly Unreliable Data. *arXiv* **2019**, arXiv:1911.04383.
46. Chenaghlou, M.; Moshtghi, M.; Lekhie, C.; Salahi, M. Online Clustering for Evolving Data Streams with Online Anomaly Detection. Advances in Knowledge Discovery and Data Mining. In Proceedings of the 22nd Pacific-Asia Conference, PAKDD 2018, Melbourne, VIC, Australia, 3–6 June 2018; pp. 508–521.
47. Firoozjaei, M.D.; Mahmoudyar, N.; Baseri, Y.; Ghorbani, A.A. An evaluation framework for industrial control system cyber incidents. *Int. J. Crit. Infrastruct. Prot.* **2022**, *36*, 100487. [CrossRef]
48. Chen, Q.; Zhou, M.; Cai, Z.; Su, S. Compliance Checking Based Detection of Insider Threat in Industrial Control System of Power Utilities. In Proceedings of the 2022 7th Asia Conference on Power and Electrical Engineering (ACPEE), Hangzhou, China, 15–17 April 2022; pp. 1142–1147.
49. Zhao, Z.; Mehrootra, K.G.; Mohan, C.K. Online Anomaly Detection Using Random Forest. In *Recent Trends and Future Technology in Applied Intelligence*; Mouhoub, M., Sadaoui, S., Ait Mohamed, O., Ali, M., Eds.; IEA/AIE 2018; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2018.
50. Izakian, H.; Pedryecz, W. Anomaly detection in time series data using fuzzy c-means clustering. In Proceedings of the 2013 Joint IFSA World congress and NAFIPS Annual Meeting, Edmonton, AB, Canada, 24–28 June 2013.
51. Decker, L.; Leite, D.; Giommi, L.; Bonakorsi, D. Real-time anomaly detection in data centers for log-based predictive maintenance using fuzzy-rule based approach. *arXiv* **2020**, arXiv:2004.13527v1.
52. Masdari, M.; Khezri, H. Towards fuzzy anomaly detection-based security: A comprehensive review. *Fuzzy Optim. Decis. Mak.* **2020**, *20*, 1–49. [CrossRef]
53. de Campos Souza, P.V.; Guimarães, A.J.; Rezenede, T.S.; Silva Araujo, V.J.; Araujo, V.S. Detection of Anomalies in Large-Scale Cyberattacks Using Fuzzy Neural Networks. *AI* **2020**, *1*, 92–116. [CrossRef]
54. Habeeb, R.A.A.; Nasauddin, F.; Gani, A.; Hashem, I.A.T.; Amanullah, A.M.E.; Imran, M. Clustering-based real-time anomaly detection—A breakthrough in big data technologies. *Trans. Emerg. Telecommun. Technol.* **2022**, *33*, e3647.
55. Mahanta, A.K.; Mazarbhuiya, F.A.; Baruuah, H.K. Finding calendar-based periodic patterns. *Pattern Recognit. Lett.* **2008**, *29*, 1274–1284. [CrossRef]
56. Mazarbhuiya, F.A.; Mahanta, A.K.; Baruah, H.K. The Solution of fuzzy equation A+X=B using the method of superimposition. *Appl. Math.* **2011**, *2*, 1039–1045. [CrossRef]
57. Zadeh, L.A. Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets Syst.* **1978**, *1*, 3–28. [CrossRef]
58. Loeve, M. *Probability Theory*; Springer Verlag: New York, NY, USA, 1977.
59. Klir, J.; Yuan, B. *Fuzzy Sets and Logic Theory and Application*; Prentice Hill Pvt. Ltd.: Upper Saddle River, NJ, USA, 2002.
60. Qiana, Y.; Dang, C.; Liaanga, J.; Tangc, D. Set-valued ordered information systems. *Inf. Sci.* **2009**, *179*, 2809–2832. [CrossRef]
61. Stripling, E.; Baeseens, B.; Chizi, B.; Broucke, B.V. Isolation-based conditional anomaly detection on mixed-attribute data to uncover workers' compensation fraud. *Decis. Support Syst.* **2018**, *111*, 13–26. [CrossRef]

62. Ding, Z.; Fei, M. An Anomaly Detection Approach Based on Isolation Forest Algorithm for Streaming Data using Sliding Window. *IFAC Proc. Vol.* **2013**, *46*, 12–17. [CrossRef]
63. Abdullah, J.; Chandran, N. Hierarchical Density-based Clustering of Malware Behaviour. *J. Telecommun. Electron. Comput. Eng. (JTEC)* **2017**, *9*, 159–164.
64. KDD CUP'99 Data. Available online: https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html (accessed on 15 January 2020).
65. Kitsune Network Attack Dataset. Available online: https://github.com/ymirsky/Kitsune-py (accessed on 12 December 2021).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.