energies

MDPI

Article

Simulation Study on the Electricity Data Streams Time Series Clustering

Krzysztof Gajowniczek [1,*], Marcin Bator [1], Tomasz Ząbkowski [1], Arkadiusz Orłowski [1] and Chu Kiong Loo [2]

[1] Department of Artificial Intelligence, Institute of Information Technology, Warsaw University of Life Sciences SGGW, 02-776 Warsaw, Poland; marcin_bator@sggw.pl (M.B.); tomasz_zabkowski@sggw.pl (T.Z.); arkadiusz_orlowski@sggw.pl (A.O.)

[2] Department of Artificial Intelligence, Faculty of Computer Science & Information Technology, University Malaya, Kuala Lumpur 50603, Malaysia; ckloo.um@um.edu.my

* Correspondence: krzysztof_gajowniczek@sggw.pl

Received: 29 December 2019; Accepted: 15 February 2020; Published: 19 February 2020

Abstract: Currently, thanks to the rapid development of wireless sensor networks and network traffic monitoring, the data stream is gradually becoming one of the most popular data generating processes. The data stream is different from traditional static data. Cluster analysis is an important technology for data mining, which is why many researchers pay attention to grouping streaming data. In the literature, there are many data stream clustering techniques, unfortunately, very few of them try to solve the problem of clustering data streams coming from multiple sources. In this article, we present an algorithm with a tree structure for grouping data streams (in the form of a time series) that have similar properties and behaviors. We have evaluated our algorithm over real multivariate data streams generated by smart meter sensors—the Irish Commission for Energy Regulation data set. There were several measures used to analyze the various characteristics of a tree-like clustering structure (computer science perspective) and also measures that are important from a business standpoint. The proposed method was able to cluster the flows of data and has identified the customers with similar behavior during the analyzed period.

Keywords: clustering; data stream; machine learning; smart metering; time series

1. Introduction

The term data stream refers to a potentially unwieldy, continuous, and rapid sequence of information. Unlike traditional data forms, which are invariable and static, the data stream has its own unique features, such as (1) it consists of a continuous flow of very large data; (2) it is rapidly evolving data that occurs in real-time with quick response requirements; (3) multiple access to the data stream is almost impossible; and (4) storage of the data stream is restricted. Currently, data streams occur in many real-world scenarios. For example, they are generated from sensors, network traffic, satellites, and other interesting use cases. They must be processed quickly and draw as much knowledge as possible. Data streams have their own specificity of data processing and exploration. They can be very fast, one cannot process the entire history of data streams in memory, so it has to be done incrementally or in (e.g., sliding) windows. One of the usual machine learning problems is data clustering, either example clustering or time-series whole clustering [1]. In those circumstances, data streams clustering has become an active research topic with applications in several contexts [2]. The aim is to group, in the same cluster, data streams that have similar properties and behavior over time (this is why it can be referred to as time series data streams clustering), while the data streams of different clusters must exhibit different characteristics [3]. Despite the proposal of many data stream

clustering techniques, only a few of them try to solve the problem of grouping data streams that consist of many time-series [4].

Exploration of data streams in subsequent time windows allows for a deep understanding of cluster behavior. Real-time data arrive in a sequential form, in which the basic distribution can change at specific intervals (this is the so-called concept of drift [5]). When processing real-time streaming data, limited memory and the fact that only one input processing run is available are particular challenges. Traditional algorithms are not suitable for this type of data as they extract patterns from the data, taking into account global properties, rather than local properties. In addition, they require a complete set of training data. For example, changes in the stream data of smart meters are expected to occur depending on many factors [6,7]. Smart grid data often present challenges such as very large sizes, high dimensionality, skewed distribution, rarity, and seasonal variations [8]. With incoming readings every 15 or 30 min, these data have specific characteristics and their analysis is difficult for the following reasons:

- they are of high dimensionality because they are usually very long;
- they have multiple seasonalities, i.e., daily, weekly, and sometimes monthly consumption patterns;
- they have annual patterns related to vacation, holidays, or seasonal change;
- there are many other stochastic factors affecting consumption, such as the weather, unexpected holidays, black-outs, market changes, special events, etc.

The above reasons bring out many challenges in discovering the segmentation of various customers based on their electricity consumption data. This would not only benefit energy service providers by providing them with demand response forecasts but would also reveal the real economic structure. Such segmentation can be used in an integrated planning system where an adequate choice in real-time among the available load management alternatives is critical to effectively meet the requirements of the system [9]. Electricity consumer consumption patterns also provide valuable information to determine optimal tariffs [8,10]. Load profiling is becoming one of the most appropriate methods for dealing effectively with customer energy consumption data.

In this article, we will discuss methods of data stream clustering. There are various implementations dedicated for this purpose, such as the stream package in R [11], Massive Online Analysis software written in Java [12] or scikit-learn library in Python [13]. Unfortunately, this software provides methods for only one stream and not multiple stream processing. However, in this research we want to show the clustering of multiple data streams, i.e., from multiple sources (e.g., sensors). To this end, we have proposed an approach to grouping multiple data stream time series in order to:

- monitor and analyze the smart grid;
- create consumer profiles (extract typical patterns of consumption);
- monitor and detect important changes (behavior) in the grid;
- unsupervised classification of consumers.

These might lead further to creating more forecastable groups of customers. The algorithm presented below is an extended version of the algorithm presented in [4] (for further details please see Section 4.1). Using smart metering data, we aim to answer the following research questions:

- Is it possible to use the stream mining approach to cluster time series of the individual electricity consumption? If so, to what extent?
- Is the standard approach for time series clustering, based on the whole data history, better? If so, to what extent is it possible to get close to these results?
- What impact on the final results do the values of the input parameters of the proposed algorithm have?

In order to answer the aforementioned research questions, we will compare two cases. The first one, considered as a ground truth/benchmark, will use the entire history of data streams (at each

iteration the analyzed period is extended without losing the history). We will refer to this "static" approach as a long-term version. The second "online" approach incorporating non-overlapping sliding windows will be denoted as a short-time version (at each iteration new period is analyzed i.e., we lose the history). The research is carried out based on the dataset provided by the Irish Commission for Energy Regulation (CER) [14], using smart meter data sampled at 30-min intervals.

The remainder of this paper is organized as follows: Section 2 provides an overview of the similar research problems for data stream time series clustering and electricity consumption segmentation. In Section 3, the theoretical framework of the proposed algorithm is presented. In Section 4, the research framework is outlined, including the details of numerical implementation, evaluation measure description, and algorithm parameter settings. Section 5 outlines the experiments and presents the discussion of the results. The paper ends with concluding remarks in Section 6.

2. Literature Review

Data stream clustering is a fast-growing part of data mining. Silva et al. [15] created a comprehensive survey on this phenomenon, which discusses the types of data stream clustering techniques and the corresponding challenges. So far, most of the attention in the literature has been paid to object-based data stream clustering that focuses on one data stream clustering. The flagship methods in this category are: ClusStream [16], D-Stream [17], DBSTREAM [18], MuDi [19], and Str-FSFDP [20]. However, we are focused on the multiple data stream (or time series stream) clustering problems (consumers in our case), therefore, we list the most important works below [15].

Dai et al. [21] presented the clustering on-demand framework (COD) for the dynamic clustering of multiple data streams. It involves a single scan of data to collect online statistics and compact multi-resolution approximations to suit the time and space constraints of the data stream environment. COD consists of two phases, i.e., the online maintenance phase and the offline clustering phase. The former provides an effective mechanism for maintaining hierarchical summaries of data streams with multiple resolutions, while the latter has been developed to find an approximation of the desired sub-streams from the hierarchy of summaries according to cluster queries [21].

Chen [22] proposed a CORREL-cluster algorithm, which is an extended version of the COD algorithm, to support real-time processing. It offers a scheme for the segmentation of time horizons and the storage of statistical information for each time segment. The authors proved that the scheme allows accurate approximations of the correlation coefficients for any time, and thus allows users to obtain clusters for data streams in the desired time window.

Beringer and Hllermeier [23] developed an efficient online version of the fuzzy C-means algorithm (Fuzzy C-Means on Data Streams; FCM-DS), which seems to be particularly useful in applications of this kind, in which the cluster structure is constantly changing. Due to the scalable online transformation of the original data, this method allows one to quickly calculate the approximate distance between the streams [23].

Rodrigues et al. [24] introduced the Online Divisive-Agglomerative Clustering (ODAC) algorithm which constantly maintains a tree hierarchy of clusters that evolves with data, using a top-down strategy. The division criterion is a correlation-based measure of similarity between time series, dividing each node by the furthest pair of streams. The algorithm also uses a merging operator that re-aggregates a previously split node to react to changes in the correlation structure between time series. Splitting and merging operators are launched in response to changes in the diameters of existing clusters, assuming that in a stationary environment the extension of the structure leads to a reduction in the diameters of these clusters [24].

Chen et al. [25] proposed a cluster algorithm (called IDEStream) for multiple data streams based on autoregressive (AR) modeling to measure correlations between data streams. The algorithm uses an estimated frequency spectrum to extract the relevant characteristics of the data streams. Each stream is represented as the sum of the spectral components and the correlation is measured component-wise. Each spectral component is described by four parameters including amplitude, phase, damping rate, and frequency. The delay correlation ε between two spectral components is calculated and used as a measure of similarity in the grouping of data streams [25].

Pereira and de Mello [4] presented the TS-Stream algorithm, which works by calculating several descriptive time series measures and building an incremental decision tree. Divisions are performed using features that distinguish between different time series models. Appropriate features are selected based on the criterion of minimizing variance. The algorithm can progressively expand or shrink the tree according to changes in the stream that change the variance of the node.

Khan et al. [8] suggested an algorithm allowing the discovery of clusters in the data of smart meters in real-time. The algorithm consists of three phases. In the first phase, it uses a gamma mixture model to identify dense units incoming data, and rare values are dissolved. During the second phase, the data from a single time window are collected and the data are grouped. Finally, the algorithm performs an incremental ensemble grouping of sets between the obtained groupings of two consecutive time windows.

Laurinec and Lucká [6,7] presented an interpretable approach for the grouping of multiple data streams in a smart grid, used to improve the forecasting accuracy of aggregated electricity consumption and grid analysis, called ClipStream. In their approach, consumer time-series streams are compressed and represented by interpretable features separated from clipped representation (online phase). The proposed representation is incremental in the sense of the window method. The offline phase consists of clustering filtered representations by the K-medoids method with the Partition Around Medoids (PAM) algorithm. Eventually, time series are aggregated within clusters to create more predictable groups of consumers.

3. Algorithm

3.1. Notations and Data Representation Model

A time series is a series of data points that are indexed (or listed or graphed) in a temporal order. Most often a time series is a sequence made at successive equally spaced points in time (e.g., measurement of electricity consumption at a given time interval, i.e., 30 min.). It is, therefore, a sequence of data of discrete-time. In the remainder of this article, $\boldsymbol{s}_j = \left\{s_{j,1}, s_{j,t}, \ldots, s_{j,n}\right\}^T$ denotes partial realizations from a j-th ($j = 1, \ldots, m$) real-valued processes $S_j = \left\{S_{j,t}, t \in \mathbb{Z}\right\}$. Formally, the problem of clustering streaming time series can be defined as follows. Let $\boldsymbol{S} = \left\{\boldsymbol{s}_1{}^T, \boldsymbol{s}_j{}^T, \ldots, \boldsymbol{s}_m{}^T\right\}$ be a data stream composed of m time series each of length n ($\boldsymbol{S}$ is a matrix with m rows and n columns). For a l-th ($l = 1, \ldots, k$) non-overlapping time windows (blocks) with w time slots (intervals), $\boldsymbol{B}_l$ is a subset (of columns) of $\boldsymbol{S}$, i.e., a matrix of dimension $m \times w$ (each block consists of a subset of times series from the same time interval. For a given block, $\boldsymbol{L}_l = \left\{L_{l_1}, L_{l_o}, \ldots, L_{l_p}\right\}$ represents a partition (of rows) of $\boldsymbol{B}_l$ such that L_{l_o} is the o-th cluster (leaf) of $\boldsymbol{L}_l$, $L_{l_o} \cap L_{l_p} = \varnothing$, $\forall o \neq p$ and $\cup_{o=1}^{p} L_{l_o} = \boldsymbol{B}_l$ [4].

Based on the above notation, a simple model of data representation is depicted in Figure 1. On the left-hand side of this figure, there is a data stream, $\boldsymbol{S}$, with m time series divided into k disjoint blocks each of length w (here w = 5). The right part of this figure illustrates an exemplary partition of the m time series from the l-th window ($\boldsymbol{B}_l$) into L_{l_p} cluster (leaf).

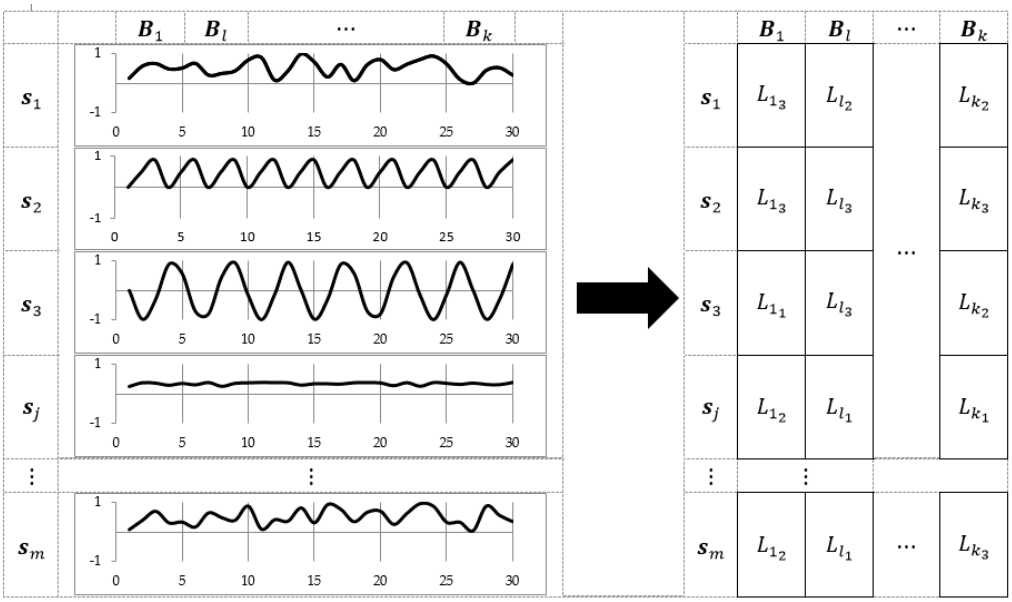


Figure 1. An exemplary data representation model with clustering.

3.2. Time Series Descriptive Measures

The algorithm (please see Algorithm 1 and Figure A9 in Appendix B) induces a model with a structure like a decision tree but built in an unsupervised manner. A top-down strategy is used to build the tree starting with all series in the same main cluster (root) and gradually creating divisions or aggregations. Each intermediate node performs a binary test of the type $value_{feature} \leq x$ on a particular feature (indices) calculated using time series descriptive measures. When the leaf is reached, the series is stored together with other series belonging to the same leaf [4].

The first step of the algorithm (please see the sixth line of Algorithm 1) is to calculate the descriptive measures (*feature* function) of each time series. This produces a *Features* matrix of dimension $m \times f$, where f denotes the number of characteristics (or coefficients). To make all features comparable to each other (it is required because the algorithm uses the variance minimization criterion), the score-z normalization is calculated ($x = (x - \mu)/\sigma$) for every column of the matrix.

A natural way to model the time series would be to use generating functions to describe their behavior over time. Unfortunately, many of the current clustering techniques do not capture different characteristics of the generating function, e.g., linearity, stationarity, or stochasticity. Therefore, the algorithm incorporates many descriptive measures to obtain appropriate characteristics of the generating function in order to better characterize the similarity between time series.

In these circumstances, Auto Mutual Information (AMI) is introduced as the first feature which is used to measure how much one random variable tells us about another variable [26]. In the context of time series analysis, AMI helps to quantify the amount of knowledge gained about the value of $s_{t+\tau}$ when observing s_t, where τ is a lag. To measure the AMI of a time series, a histogram (with bins) of the data should be created. Let p_i be the probability that the signal has a value inside the i-th bin, and let $p_{ij}(\tau)$ be the probability that s_t is in bin i and $s_{t+\tau}$ is in bin j. Then, the AMI for time delay, τ, is defined as [4]:

$$AMI(\tau) = \sum p_{ij}(\tau) \log\left(\frac{p_{ij}(\tau)}{p_i p_j}\right) \quad (1)$$

Depending on the base of the logarithm used to define the *AMI*, the *AMI* is measured in bits (base 2, also called Shannons), nats (base e), or bans (base 10, also called Hartleys).

The second investigated measure is Hurst exponent which is a measure of long-term memory of the time series [27]. It relates to the autocorrelations of the time series and the rate at which these decrease as the lag between pairs of values increases. There are different ways of estimating

the exponent; most often the Scaled Range approach is taken into account. The Hurst exponent, H, is defined in terms of the asymptotic behavior of the Rescaled Range as a function of the time span of a time series, as follows [4]:

$$\frac{R_t}{S_t} = ct^H \tag{2}$$

where c is a constant, t is the time span of the observation (number of data points in a time series), R_t is the range of the first t cumulative deviations from the mean, and S_t is their standard deviation. Typically, Hurst's exponent determines the scale of self-similarity and properties of the correlation between Brown's fractional noise and Gauss' fractional process. In the case of self-similar processes, the statistical characteristics do not change for different levels of aggregation. The Hurst exponent is referred to as the "index of dependence" or "index of long-range dependence" [4]. It quantifies the relative tendency of a time series either to regress strongly to the mean or to cluster in a direction [28].

A value H in the range 0.5–1 indicates a time series with long-term positive autocorrelation, meaning both that a high value in the series will probably be followed by another high value and that values a long time into the future will also tend to be high. A value in the range 0–0.5 indicates a time series with long-term switching between high and low values in adjacent pairs, meaning that a single high value will probably be followed by a low value and that the value after that will tend to be high, with this tendency to switch between high and low values lasting a long time into the future. A value of $H = 0.5$ can indicate a completely uncorrelated series, but in fact, it is the value applicable to the series for which the autocorrelations at small time lags can be positive or negative but where the absolute values of the autocorrelations decay exponentially quickly to zero [4].

The third measure that is used in this article is the Discrete Fourier Transform (DFT) [29] describing time series in the frequency domain. This measure converts a finite sequence of equally-spaced samples of a function into a same-length sequence of equally-spaced samples of the Discrete-Time Fourier transform (DTFT). Therefore, according to the authors in [4], it splits the time series into the basic frequencies by a combination of sinusoidal waves.

We used these measures to obtain different characteristics of time series generating functions, i.e., the DFT helps to identify nonlinearities occurring in time series by modeling them as a combination of sinusoidal functions. The Hurst exponent helps to determine the stochasticity of a series, while AMI helps to identify deterministic relationships between past and future observations [4].

3.3. Finding Significant Features

To start dividing a series into different nodes/clusters (and eventually leaves), each time the *SignificantBestFeature* algorithm is called (please see line 9 and 34 of Algorithm 1) it is responsible for finding the best feature for the binary test of the current node [4]. This algorithm takes as an input normalized *Features* matrix of dimension $m \times f$ derived in the sixth line of the main algorithm. In general, this procedure, similar to the CART or C4. 5 tree-based algorithms [30], involves minimizing the weighted variance criterion (based on the features) of the form:

$$Gain = \sigma^2(V) - \frac{n_{left} * \sigma^2\left(V_{left}\right) + n_{right} * \sigma^2\left(V_{right}\right)}{n} \tag{3}$$

where $\sigma^2(\cdot)$ is the variance function, V is the current node with n time series, and V_{left} and V_{right} are the nodes created after the division, each consisting of n_{left} and n_{right} time series, respectively. If there is at least one descriptive measure capable of separating time series according to their generation functions (meaning that time series comes from the same mathematical), then after the partition, each child node will contain series with a smaller variance i.e., similar values for the same feature [4].

The above algorithm requires a β parameter for indicating features that do not contribute to the separation of time series. This is done by calculating Shannon's entropy (each feature is binned into predefined h bins [31]) of the descriptive measures, therefore the *SignificantBestFeature* algorithm shall

recognize the features as enforceable only if their entropy is greater than β. If, for example, a particular feature is split into two bins ($i = 1, 2$), in both extremes, when $p_i \approx 0$ or $p_i \approx 1$, one of the compartments has an observable probability close to 1, because $p_1 + p_2 = 1$. These cases are not as useful as each time series has the same value for a descriptive measure, therefore, they cannot be separated into different clusters [4].

Finally, the function, *SignificantBestFeature*, provides the best division that maximizes gain by applying a weighted variance criterion (Formula 3) and assessing all possible cuts of all available features. Having this kind of information, whenever possible (see line 13 and 42), the algorithm splits the current node into two child nodes.

3.4. Node Splitting and Aggregation

In each iteration, after receiving a new data batch $\boldsymbol{B}_l$ (and calculating new feature matrix), the algorithm groups series according to the current tree model (line 17 of the algorithm); in other words, the structure obtained in the previous iteration. Then the update phase begins, in which the splits and/or aggregations are checked and executed if possible/necessary [4]. Before deploying the Algorithm 1, it is required to set values of the parameters $\alpha \in [0, 1]$, $\lambda \in [0, 1]$ and *minSeries*.

Algorithm 1: Clustering algorithm.

Input: Time series data stream $\boldsymbol{S} = \left\{{\boldsymbol{s}_1}^T, {\boldsymbol{s}_j}^T, \ldots, {\boldsymbol{s}_m}^T\right\}$ of length m, window length (w), tuned parameters ($\alpha, \beta, \lambda, minSeries$)
Output: Tree ($\boldsymbol{T}$) with clustered time series data streams

/1/ $\boldsymbol{T} \leftarrow \varnothing$
/2/ divide $\boldsymbol{S}$ into k non-overlapping blocks of length w; $\boldsymbol{B} = \{\boldsymbol{B}_1, \boldsymbol{B}_l, \ldots, \boldsymbol{B}_k\}$
/3/ **for each** $\boldsymbol{B}_l \in \boldsymbol{B}$ **do**
/4/ clear leaves (remove series from the previous block $\boldsymbol{B}_l$, preserve the structure of the tree $\boldsymbol{T}$)
/5/ **for each** $\boldsymbol{s}_j \in \boldsymbol{B}_1$ **do**
/6/ $Features[j] = features\left(\boldsymbol{s}_j\right)$
/7/ **end**
/8/ **if** $\boldsymbol{T} = \varnothing$ **then do**
/9/ $SignificantBestFeature =$ find significant features based on the *Features* and β
/10/ **if** $SignificantBestFeature = \varnothing$ **then do**
/11/ continue // next $\boldsymbol{B}_l$
/12/ **end**
/13/ $LeftChild, RigntChild = Split(\boldsymbol{T}, SignificantBestFeature, FeatureS)$
/14/ $\boldsymbol{T} = \boldsymbol{T} \cup \{LeftChild, RigntChild\}$
/15/ create set of leaves; $\boldsymbol{L}_l = \left\{L_{l_{1(LeftChild)}}, L_{l_{2(RigntChild)}}\right\}$
/16/ **else**
/17/ cluster time series according to the current $\boldsymbol{T}$ structure
/18/ create set of leaves based on the current $\boldsymbol{T}$ structure; $\boldsymbol{L}_l = \left\{L_{l_1}, L_{l_o}, \ldots, L_{l_p}\right\}$
/19/ **end**
/20/ **while** $\boldsymbol{L}_l \neq \varnothing$
/21/ $curLeaf =$ first from $\boldsymbol{L}_l$
/22/ $\boldsymbol{L}_l \backslash \{curLeaf\}$
/23/ $\boldsymbol{aggLeaf} = FALSE$
/24/ **if** $\#curLeaf < minSeries$ **then do**
/25/ $\boldsymbol{aggLeaf} = TRUE$
/26/ **else**
/27/ VP = calculate Variance of the parent node of the $curLeaf$ based on the *Features*
/28/ WVC = calculate Weighted Variance of children based on the *Features* using $curLeaf$ and sibling of the $curLeaf$

/29/ **if** $WVC > \lambda * VP$ **then do**
/30/ ***aggLeaf*** $= TRUE$
/31/ **end**
/32/ **end**
/33/ **if** ***aggLeaf*** $= TRUE$ **then do**
/34/ $Parent =$ subtree starting from the parent node of the $curLeaf$
/35/ delete from $\boldsymbol{L}_l$ all leaves below the $Parent$
/36/ $AggregateToLeaf(Parent)$
/37/ **end**
/38/ ***splitedLeaf*** $= FALSE$
/39/ $SignificantBestFeature =$ find significant features for the $curLeaf$ based on the $Features$ and β
/40/ **if** $SignificantBestFeature = \varnothing$ **then do**
/41/ VP = calculate Variance of the $curLeaf$ based on the $Features$
/42/ $LeftChild, RigntChild = Split(\boldsymbol{T}, SignificantBestFeature, FeatureS)$
/43/ **if** $LeftChild \geq minSeries$ and $RightChild \geq minSeries$ **then do**
/44/ WVC = calculate Weighted Variance of the $LeftChild$ and $RightChild$
/45/ **if** $WVC < \alpha * VP$ **then do**
/46/ ***splitedLeaf*** $= TRUE$
/47/ **end**
/48/ **end**
/49/ **if** ***splitedLeaf*** $= TRUE$ **then do**
/50/ $\boldsymbol{T} = \boldsymbol{T} \cup \{LeftChild, RightChild\}$
/51/ **else**
/52/ $AggregateToLeaf(curLeaf)$
/53/ **end**
/54/ **end**
/55/ **end**
/56/ **return** $\boldsymbol{T}$

Two sibling leaves ($LeftChild$ and $RightChild$) shall be aggregated (using $AggregateToLeaf$ function) if their weighted variance (WVC) is greater than or equal to λ times the variance of the parent node (VP) calculated on the basis of its test feature (see also Equation (3) and lines 26–32). This prevents maintaining splits/nodes that do not provide a reasonable reduction of the variance. Thanks to that, the structure of the tree becomes simpler and robust to noise/outliers. In this step more than two nodes/leaves can be aggregated (lines 34–36). If a particular leaf should be aggregated with its sibling (e.g., at level 3), it might happen that this sibling has its own children (at level 4). Recursively, those children also might have their own offspring at level 5, and so on. After this procedure is done, the parent node (at level 2) is again marked as a leaf. Lower values of λ make aggregation easier [4].

In the absence of aggregation, the algorithm checks for possible leaf breakdowns. In order to do this, $SignificantBestFeature$ is called and the leaf is split if the weighted variance of its possible siblings decreases by at least α times its own variance (lines 45–47). Therefore, the higher the α value, the less the variance will have to be reduced for the split [4].

Finally, the parameter *minSeries* controls the depth/complexity of the tree by preventing the split if two possible children will have less than a particular % of the total observations.

4. Research Framework and Settings

4.1. Numerical Implementation

As presented below, numerical experiments were prepared using *R* programming language working on Ubuntu 18.04 operating system on a personal computer equipped with Intel Core i7-9750H 2.6 GHz processor (12 threads) and 32 GB RAM. All results were obtained based on the modification

of the TS-Stream algorithm presented in [4]. First, the original algorithm does not have the *minSeries* parameter, which means that a particular leaf can contain only one time series (the tree can be very big and deep). Secondly, for each block, $\boldsymbol{B}_l$ the original algorithm performs only one division or aggregation of a particular node or leaf (line 20 of the Algorithm 1). In other words, if the algorithm splits a node into two children, the algorithm finishes its operation for this part of the tree, despite the fact that it could further check the possibility of further division of these two children. It results in the fact that the original algorithm needs some time to start and reach the full size. In our case, we used a different loop, which means that the tree immediately finds the right size.

4.2. Tree Characteristics and Clustering Evaluation Measures

In order to analyze results presented in Section 5 for each data stream block, $\boldsymbol{B}_l$, the following characteristics have been derived:

- **Number of leaves/clusters** (i.e., nodes that have no children);
- **Depth of a tree,** which is the distance between a leaf node from the deepest level and the root;
- **Diameter of a tree,** which is the length of the longest path between any two nodes in a tree passing through the root;
- **Degree of a tree imbalanced:** A balanced tree is a kind of a tree where for every subtree, the maximum distance from the root to any leaf is, at most, bigger by one more than the minimum distance from the root to any leaf. In this case, we report the difference between the maximum and the minimum distance among all leaves. For example, if the minimum distance is 3 and the maximum is 5, then this measure takes 2, while a balanced tree takes 0;
- **Minimal, average, median, and maximal** number of time series across all clusters/leaves;
- **Weighted volatility** of time series for a given block $\boldsymbol{B}_l$. After the partition time series are spread across several clusters. It is assumed that the volatility (measured in standard deviation) of electricity consumption in each cluster is supposed to be smaller than the variability of the time series contained just in one cluster (root). Moreover, due to the difference in the size of each leaf, this measure takes this fact into account by assigning smaller weights to a smaller leaf—similarly to the right part of Equation (3):

$$\text{Weighted volatility}_{\boldsymbol{B}_l} = \sum_{L_{l_o} \in \boldsymbol{L}_l} \frac{\#L_{l_o}}{m} * \sigma\left(L_{l_o}\right) \tag{4}$$

  where $\#L_{l_o}$ is the number of time series for a given leaf, m is the number of time series in a block $\boldsymbol{B}_l$ and $\sigma(\cdot)$ is the standard deviation of all times series assigned to a given leaf L_{l_o}.
- **Percentage of time series that do not change the cluster (same leaf) or intermediate node**. According to the left-hand side of Figure 2, let us assume that for the block $l = 2$, the tree has two clusters/leaves, i.e., L_{2_1} (offspring of the root) and L_{2_2} (offspring of the root). Next, for the block $l = 3$ the tree has three clusters, L_{3_1} (offspring of the root) and two new siblings of the former cluster L_{2_2} (now denoted as Node_{3_2}), i.e., L_{3_2} and L_{3_3} (the right part of Figure 2). If a certain time series belonged to the cluster L_{2_1} for the second block and then remained in the third cluster after the third iteration, then one can say that this time series did not change cluster (cluster L_{3_1}) or node assignment (cluster L_{3_1}). Now we assume that a particular time series was (for block $l = 2$) in leaf L_{2_2} and, during the next iteration, changed its affiliation to cluster L_{3_2}. One can say that this time series did change cluster and simultaneously, did not change node assignment as leaf L_{2_2} from the previous iteration is now the parent node of L_{3_2} and L_{3_3}. This implicates that this measure will always be greater for the node assignment case;
- **Percentage of time series that stayed in the same cluster.** One can say that two times series "travel together" if they were together in the same leaf, let us say L_{2_1} and then during the next iteration remained together in any other cluster e.g., L_{3_2} or L_{3_3}.

- **Percentage of clusters/leaves that have remained, have been deleted or have been created** in the next iteration;
- **Adjusted Rand Index (ARI),** which is the corrected-for-chance version of the Rand index [32] measuring the similarity between two data clusterings. This index has zero expected value in the case of random partition and is bounded above by 1, in the case of perfect agreement between two partitions. This measure requires two vectors as an input, and in this research, the first one can be considered as having true class labels, similar to the supervised learning, and the second one as predicting a class from the new instances. This measure will be used to a) derive similarity between blocks and 2) assess the similarity between the long-term and short-term approach.
- **V-measure,** which is an entropy-based measure that explicitly measures how successfully the criteria of homogeneity and completeness have been satisfied. This measure is computed as the harmonic mean of distinct homogeneity and completeness scores [33].

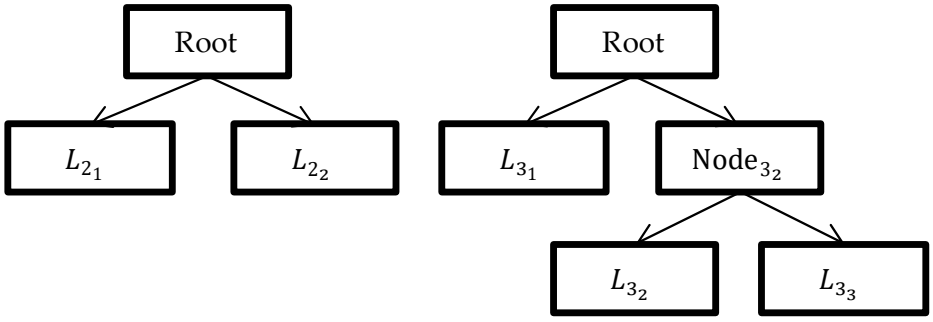


Figure 2. Exemplary tree structures from the two consecutive blocks.

All the aforementioned measures allow for the analysis of various characteristics of a tree-like clustering structure (which is important from the computer science perspective), as well as the grouping results (which are important when taking into account the business point of view).

4.3. Algorithm Parameters Setting

Before running the simulation, it is important to set the algorithm parameter values appropriately. Since the parameters α and λ have a similar influence, it is not recommended to set one value as a function of the other. During the research preparation stage, it was observed that setting these two parameters to values smaller than 0.6 resulted in almost no splits. On the other hand, values greater than 0.6 could result in a too wide and too deep tree, as in the case of the original implementation [4]. In order to prevent this issue, this implementation incorporates an additional parameter, *minSeries*, which is responsible for controlling the size of a tree, as standard implementation could provide trees with leaves having only one time series. Under these circumstances, we set the α and λ parameters to 0.6 while *minSeries*, in this article, is set to 1% and 5% (i.e., a 10 or 50 time series). This allows us to build a relatively large and extensive tree ensuring that the main dependencies in the data are found, rather than local outliers. Parameter β (indicating features that do not contribute to the separation of time series), according to the authors of the implementation of the standards, was set to 0.2.

Eventually, in order to capture monthly and mid-monthly changes in the underlying structures, the window length w of each non-overlapping block $\boldsymbol{B}_l$, has been set to 30 days and 14 days. As the electricity consumption data that are used in this article (see Section 5.1) were recorded at 30-min intervals, each window, of length 1440 or 720, records (2×24 h $\times$ 30 days) or (2×24 h $\times$ 14 days), respectively.

4.4. Benchmarking Methods

In order to compare and to assess the quality of the proposed approach, the following benchmarking methods have been used. The first one is the standard TS-Stream algorithm presented in [4] (please see

changes described in Section 4.1). The second one is the approach that utilizes the following steps. For each block $\boldsymbol{B}_l$ (of length w) the similarity between the time series is obtained using the Pearson correlation coefficient [16,24]:

$$r_{s_j s_m} = \frac{\sum_{i=1}^{w}\left(s_{ji} - \bar{s}_j\right)\left(s_{mi} - \bar{s}_m\right)}{\sqrt{\sum_{i=1}^{w}\left(s_{ji} - \bar{s}_j\right)^2}\sqrt{\sum_{i=1}^{w}\left(s_{mi} - \bar{s}_m\right)^2}} \quad (5)$$

where $\bar{s}$ is the sample mean for j-th time series from each block $\boldsymbol{B}_l$. Then, the distance matrix $\boldsymbol{D}$ is calculated as $\boldsymbol{D} = 1 - \left(r_{s_j s_m}\right)$. In the last step based on the $\boldsymbol{D}$ matrix, time series are clustered using Ward's criterion applied for the hierarchical algorithm (*hclust* function) [1]. The resulting dendrogram is cut based on the *cutree* function into groups of a number corresponding to the number of groups obtained on the basis of the proposed algorithm.

5. Empirical Analysis

5.1. Data Characteristics

This research was conducted based on the Irish Commission for Energy Regulation (CER) data set. The CER initiated a Smart Metering Project in 2007 with the purpose of undertaking trials to assess the performance of Smart Meters and their impact on consumer behavior. The project contains measurements of electricity consumption gathered from 4182 households between July 2009 and December 2010 (75 weeks in total with 30 min data granularity) [10]. However, in this study, due to the missing data in the time series, we have sampled 1000 households.

As depicted in Figure 3, for the time series aggregated for 1000 entities a number of annual, weekly, and daily seasonal cycles were observed. For instance, the daily load curves have different shapes depending on the day (workday, Saturday, Sunday, or holiday) and the season.

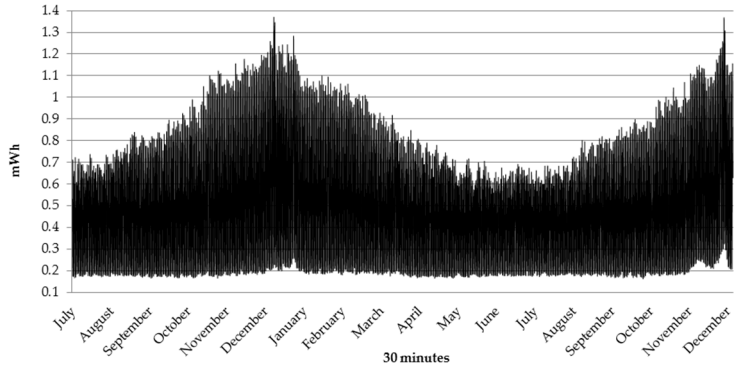


Figure 3. Thirty minutes of load data for 1000 customers recorded between 15 July 2009 and 31 December 2010.

Figure 4 presents a weekly profile with relatively low electricity consumption during the night, clearly defined peaks in the evenings, and slightly smaller peaks in the late morning. The consumption is slightly lower during the weekend days as compared to working days [10]. Moreover, to analyze the 30-min volatility, a box and whisker plot was prepared using load data for the whole period, see Figure 5 for details.

The whiskers denote the minimum and the maximum value for each hour and the box contains 50% of the data (bottom edge reflects the first quartile and top edge third quartile, while the line in the middle of the box is the median). As shown in Figure 5, the consumption during night and morning hours (0:00–7:00) usually ranges between and 0.15 MWh and 0.35 MWh every 30 min, and the volatility is rather low. On the other side, volatility during working and evening hours (10:00–24:00) is very high and ranges between 0.15 MWh and 0.35 MWh every 30 min [10].

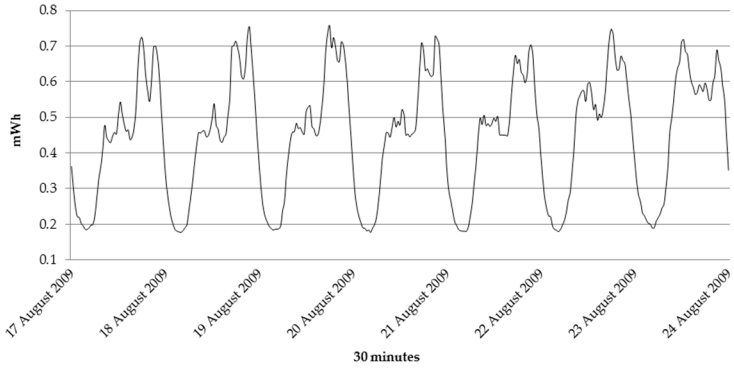


Figure 4. Daily dynamics of the 30-min load data observed between 17 August (Monday) and 24 August (Sunday) 2009.

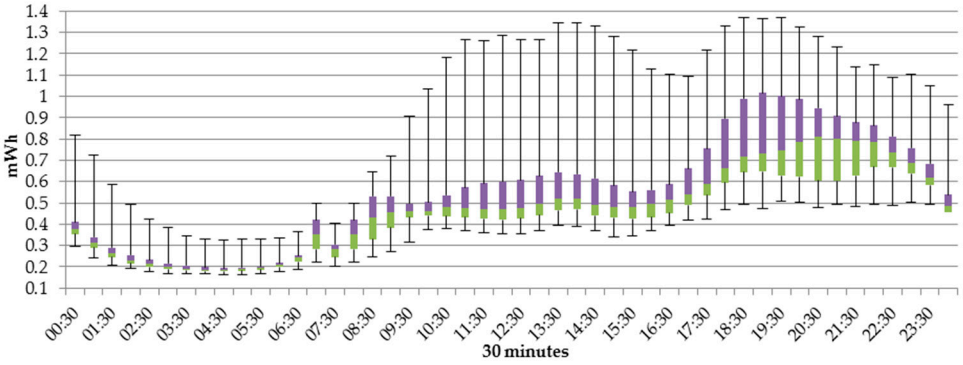


Figure 5. Thirty-minute consumption volatility in the analyzed period (15 July 2009–31 December 2010).

5.2. Research Results

This section refers to the application of the clustering algorithm mentioned in Section 3. For the sake of clarity and synthesis, the results provided below (Figure 6) are visualized only for the *minSeries* parameter set to 5% (50 time series) and window length w set to 30 days (1440 records). However, in the appendix section (Appendix A), we present the detailed results for each investigated case.

As far as the summary results are concerned, each figure is prepared in the same manner/order. There are 16 different sub-figures (marked with the letter a–u) plotting measures (for the final tree structure), as described in Section 4.2 for each data stream block $\boldsymbol{B}_l$ (in this case $l = 1, \ldots, 17$). The long-term approach, also denoted as a static version, is marked with a straight orange line, while the short-term approach ("online" version) is marked with a straight blue line. According to the first upper left Figure 6a, the long-term version usually has more leaves/clusters than the "online" approach. The opposite situation can be observed for blocks numbers 2, 5, and 13. In terms of the tree depth (Figure 6b), only for two blocks (2 and 5) does the long-term version have fewer levels. Almost the same situation can be observed in terms of the diameter of the tree (Figure 6c) and the balance of the tree (Figure 6d) where smaller (or equal) values are observed for the second and fifth block. By submitting the aforementioned information, it can be stated that the short-term version usually produces smaller, shallower, and more balanced trees with less diameter.

In terms of the different statistics of the leaves, it should be remembered that the parameter *minSeries* has great influence here. That is why the minimal number of time series, in this case, is always greater than 50 (Figure 6e). Only for a particular block where a smaller tree is produced is this statistic bigger (second block for the long-term version and eleventh block for the short-time version). The average (Figure 6f) and median (Figure 6g) number of time series in the cluster show similar behavior over time. Only at the beginning of the period (blocks 2–4) patterns have a clearer difference. A similar relationship as for the minimal number can be observed for the maximal number of the time

series (Figure 6h). Eventually, it should be noted that these four statistics have strong dependences with the four previously mentioned measures describing the complexity of each tree.

Since all of the below-described measures are computed as a comparison of two adjacent (successive) data stream blocks, results start from the second iteration. The percentage of time series that do not change the cluster (Figure 6i) for short-time version varies from 10% to 30% while the long-term version is more volatile i.e., ranging from 0% up to 45%. Noticeable is the fact that after the eighth block, the percentage of remaining time series in the same leaf is greater for the static version.

After this period, this approach has learned the underlying data structure, meanwhile, the "online" version takes into account only the local characteristics that are relatively stable over time. Considering the case when time series remains at the same common node level (Figure 6j), similar behavior is observed, however, without such significant differences at the beginning. Finally, the Figure 6k plotting the percentage of time series that stayed in the same cluster (ranging from 0% even up to 30%) allows us to conclude that it is possible to distinguish a core time series (with similar characteristics) creating a particular cluster.

Analyzing the next three Figure 6l–n describing the percentage of leaves that have remained, have been deleted, or have been created, it can be observed that until the seventh iteration, the long-term approach usually deleted clusters created based on the previous data stream block (percentages between 60% to 100%) and in their place created completely new and even more clusters (percentages even up to 300%). After the eighth block, both approaches show similar and relatively stable patterns.

Finally, analyzing the results that are important from the business perspective, we conclude that both approaches give relatively similar results (the last two Figure 6o,u at the bottom). The weighted volatility of the partitions (measured in standard deviations) reveals that for some blocks, the short-term version provides even better grouping than the long-term version. Only for block number 10 is the difference very visible; at other times, the results are almost imperceptible (at this level of complexity and data variation).

In Figure 7, we present the Adjusted Rand Index, which is a measure of the coexistence of time series in the same cluster. Bearing in mind that for such a large number of time series a coexistence can be just a coincidence, the ARI was also calculated for random partitioning (benchmark partitioning; worst case scenario partitioning) based on the distribution of the time series across all leaves in the tree. In other words, if for a particular data block a tree has three leaves, having say a 300, 500, and 200 time series, then a tree with random partitioning also has three leaves, having a 300, 500, and 200 time series, but this time with random assignments.

The maximum value of the ARI results from the random assignment were treated as reference values to judge the level of extraction similarity from the data. It is not surprising that for the long-term analysis (the right part of the figure; Figure 7b), the ARI has bigger values than for the short-term version. The first fact is that the long-term blocks overlap. The second fact is that long-term analysis uses more data. However, in this paper, the more interesting question is whether the short-term approach also allows the extraction of the underlying structure of the data. In most cases, the ARI for the short-term is at least twice the ARI for the benchmark version. This gives a positive answer on the main question stated in the paper.

In the analysis, the ARI was computed across all blocks. This means that the first block was compared with the second, the third and so on. As a result, a triangle matrix was created. Closer to the matrix diagonal, the time distance between blocks is shorter. Dark gray values denote situations when the ARI values are 10 times greater than the benchmarking, gray cells mark situations when the values are between two times and 10 times, and light-gray shows situations when values are at least bigger than the biggest random value.

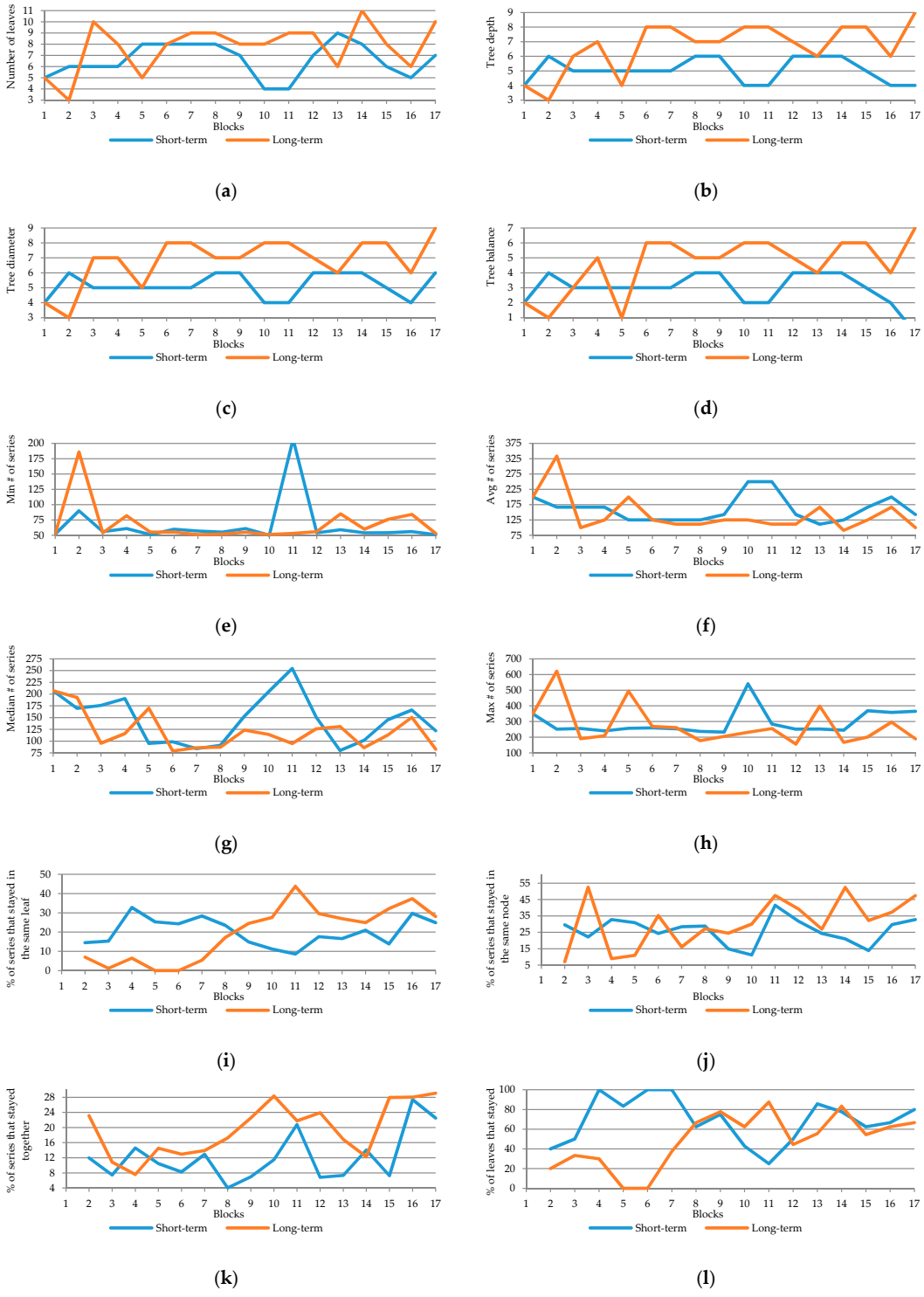


Figure 6. *Cont.*

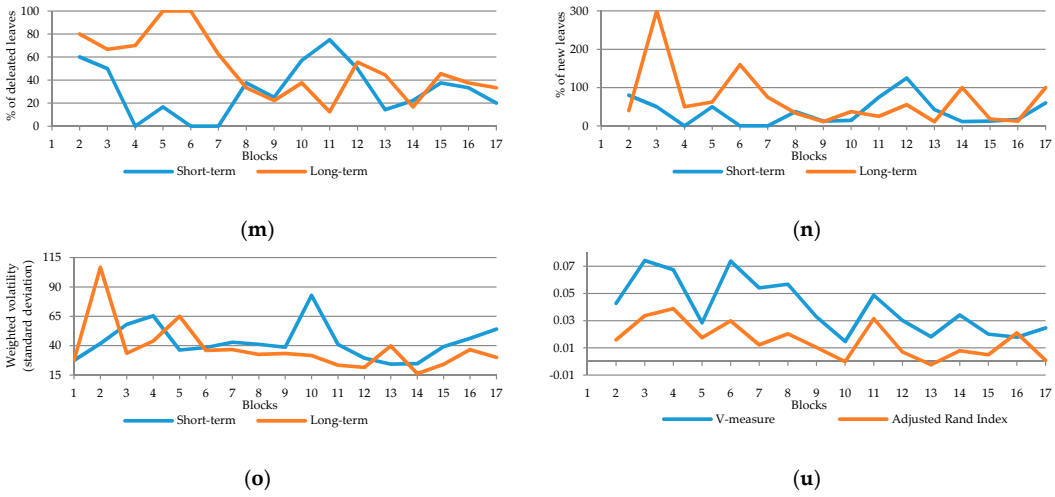


Figure 6. Detailed results for *minSeries* set at 5% and window length w set at 30 days.

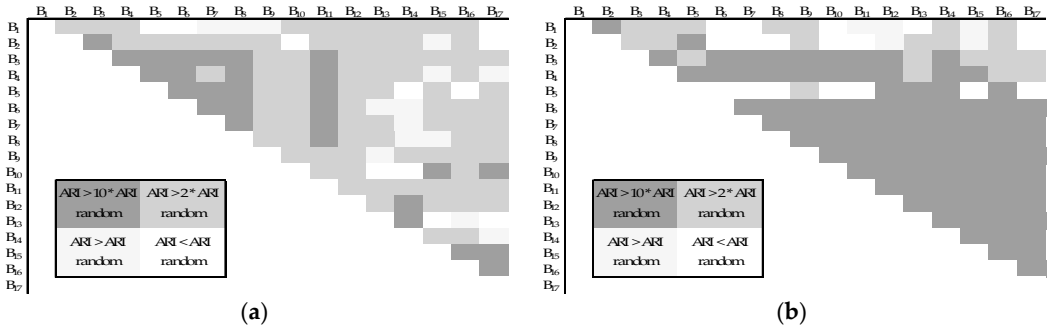


Figure 7. The Adjusted Rand Index (ARI) results for *minSeries* set at 5% and window length w set at 30 days (short-term version is presented on the left part (**a**) while long-term on the right part of the figure (**b**)).

Figures A7 and A8 (see Appendix A) present the detailed results for two benchmarking methods i.e., the basic TS-Stream algorithm and hierarchical clustering algorithm incorporating Ward's criterion and distance matrix based on the Pearson correlation coefficient. As stated before, the original TS-stream algorithm has some drawbacks. First, it performs only one division or aggregation of a particular node or leaf and a particular leaf may contain only one time series. Due to that, in Figure A7 it can be seen that the number of leaves "linearly" increases from two (for the first block) up to 108 (for the last block) or leaves consist of only one time series. After each block, the tree gets bigger but unfortunately, the percentage of series that stayed together is almost 0 which means that this algorithm is not able to discover a core time series creating a particular cluster. For hierarchical clustering, as some measures are the same as for our proposed algorithm (see Section 4.4), Figure A8 consists of fewer sub-figures. First, it can be observed that usually fewer time series stay in the same node/leaf. Secondly, according to the weighted volatility measure, this benchmarking method leads to worse clustering performance than our proposed algorithm.

6. Conclusions

Grouping of data streams is one of the most common ways of analyzing data that is potentially infinite and evolves over time. Despite the fact that the literature provides some methods of grouping whole data streams, unfortunately, most of them often deal with data streams composed of one attribute, and in the worst case, they do not apply appropriate strategies for multidimensional data streams [3].

In this work, we have presented an algorithm incrementally constructing the tree-like hierarchy of clusters from a divisive point of view, which is able to adapt to changes (shrink or expand) in

time series data streams (concept drift). The data stream batches are processed on receipt, using a single data scan. To assess the similarity between time series within each data stream, the algorithm incorporates three descriptive measures to obtain appropriate characteristics of the generating function (Auto Mutual Information, Discrete Fourier Transform, and Hurst exponent).

Although electricity consumer targets are often based on purely monetary benefits, electricity suppliers benefit from the awareness of consumers' profiles. This makes it possible to create individualized means targeting consumers with congruous usage profiles and socio-economic behaviors. Our analysis showed that there are noticeable differences between consumers' behaviors, allowing us to distinguish homogeneous groups. In order to meet these challenges and to balance the system, customer profiling seems to be the solution to the problem of instability in power systems [34].

With our analysis, we confirm that partitioning the customers into disjoint segments based on short-term behavioral usage characteristics (obtained using descriptive features) is feasible and can be achieved with reasonable accuracy compared to the long-term approach (first and second research question). This statement is supported by the results related to the Adjusted Rand Index (matrix-like figures in Figure 6 and in Appendix A Figures A4–A6) and weighted volatility measure (please see middle-right-hand part of Figure 7 and Appendix A Figures A1–A3). Eventually, the third research question regarding the influence of the input parameters on the final results is shown primarily on the right-bottom part of Figure 7 and Appendix A Figures A1–A3 (single line for the Adjusted Rand Index) and is based on a comparison of other figures showing the clustering tree structure characteristics.

Our analysis has shown the usefulness of using such an approach for electricity time series clustering. However, there are other real-word applications e.g., the stock market. Let us suppose that an investor wants to choose the best stocks to create a portfolio. Ideally, it should be diversified and composed of companies from several economic segments. Diversification helps to compensate for the eventual losses from a sector with profits in others, minimizing the risk margin. Grouping data streams consisting of stock price time series can help the investor understand the relationship between different stocks and build a portfolio by selecting them from several clusters. Due of the mass volume of banking transactions and customer information, an Anti-Money Laundering and Countering the Financing of Terrorism (AML/CFT) policy is inconceivable without the algorithms that allow the automatic filtering of transactions and entities under embargo, as well as the ability to profile and scan customer transactions against AML/CFT scenarios configured by each institution (suspicious behavior). With adequate tools, it is possible for the banking sector to analyze transactions very quickly and to cross-check them, for example, with geolocation data. They can also increase the reliability of customer data analysis by methods resistant to absent data or data that has not been updated in customer records.

Future research will be focused on developing a completely scalable system (providing interpretable results) for a large number of time series within the data stream, in the presence of:

- a variable number of sensors (some sources are removed while others are newly created);
- missing or heterogeneous recordings;
- concept drift of different kinds, such as sudden, gradual, incremental, or recurring.

Due to that, we will investigate different incrementally computable time series similarity measures, allowing us to incorporate the past knowledge of the underlying structure of the data (previous structure of the tree). The system will use the dissimilarity measures supporting splits or aggregations of the clustering tree on a significance level given by the Hoeffding bound [35]. In the future, we will investigate the influence (sensitivity of the algorithm) of the input parameters on the final results.

Author Contributions: K.G. prepared the simulation and analysis and wrote the Sections 1–6 of the manuscript; M.B. wrote Section 1, Section 5, and Section 6; T.Z. wrote Section 1, Section 2, and Section 6 of the manuscript; A.O. and C.K.L. coordinated the main theme of the research. All authors read and approved the final manuscript.

Funding: This research received no external funding.

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

Appendix A

Figure A1. Detailed results for *minSeries* set at 1% and window length *w* set at 30 days.

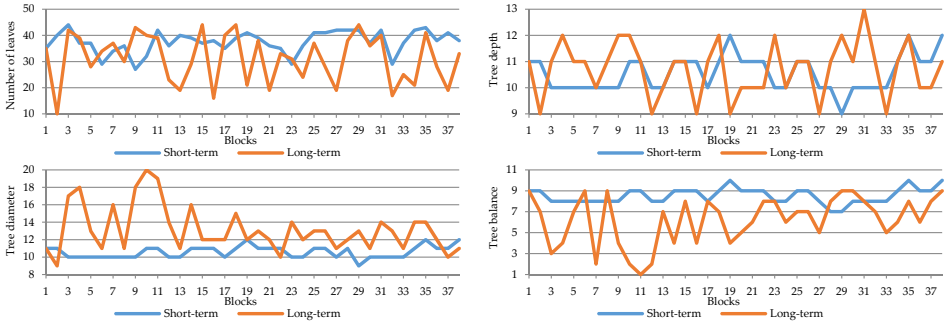


Figure A2. *Cont.*

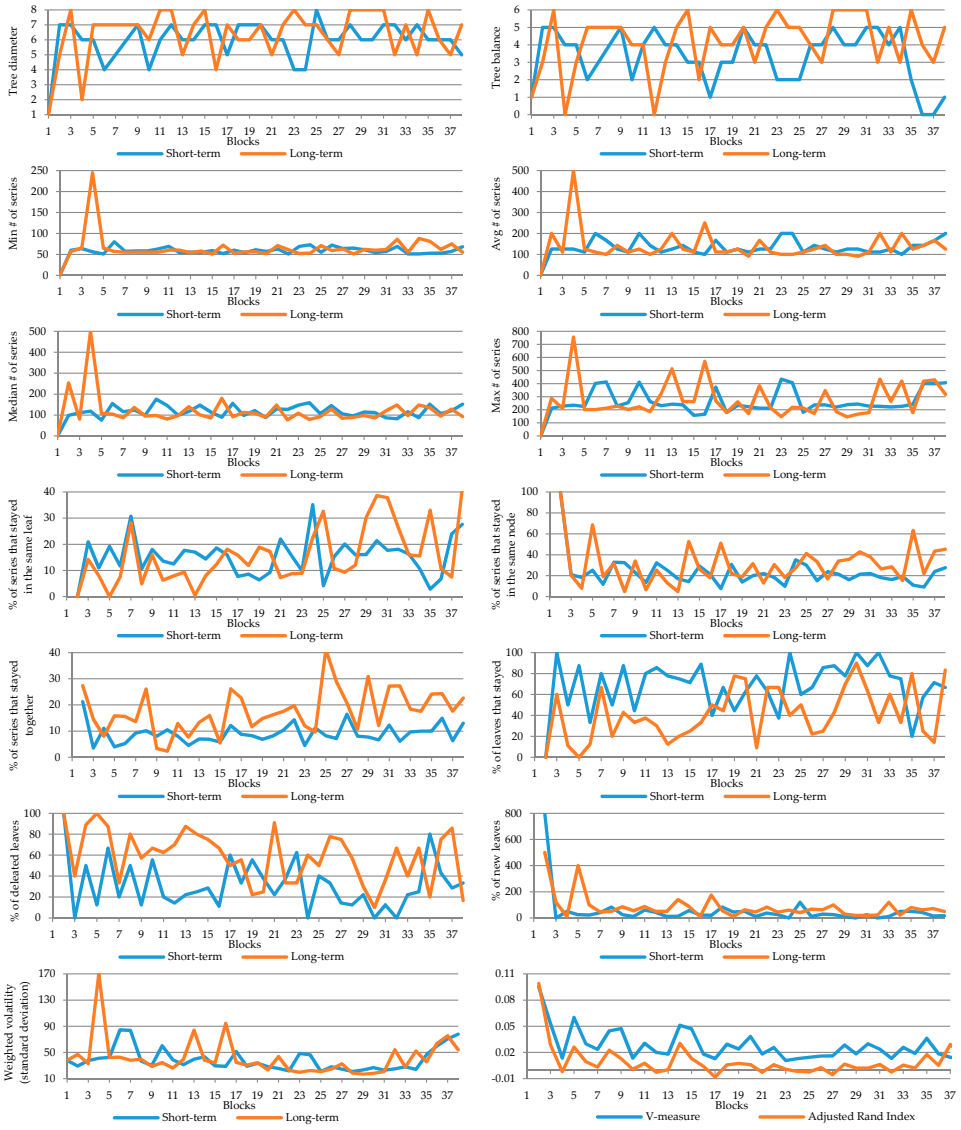


Figure A2. Detailed results for *minSeries* set at 5% and window length *w* set at 14 days.

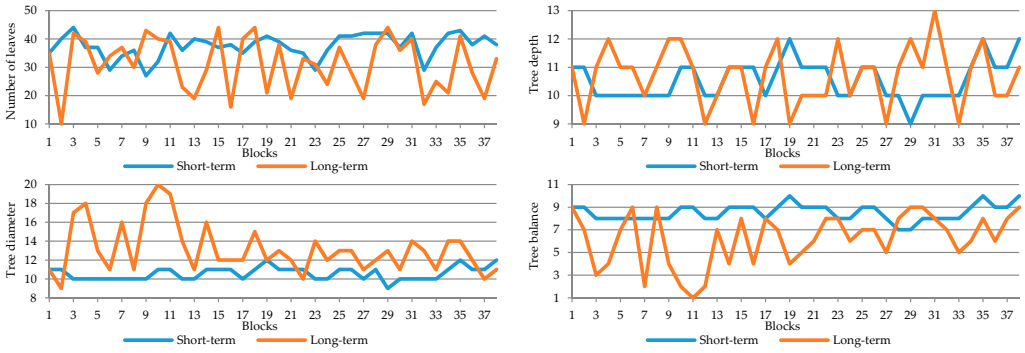


Figure A3. *Cont*.

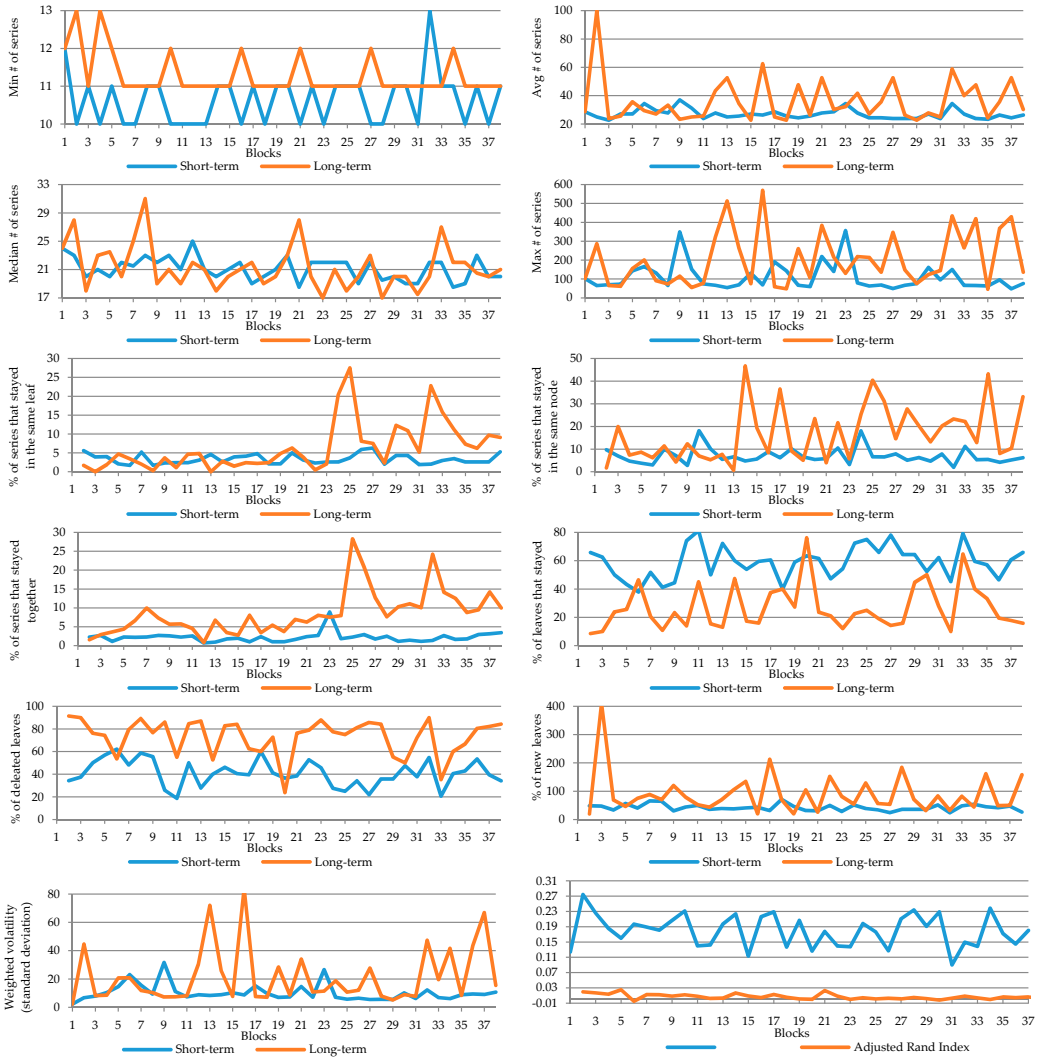
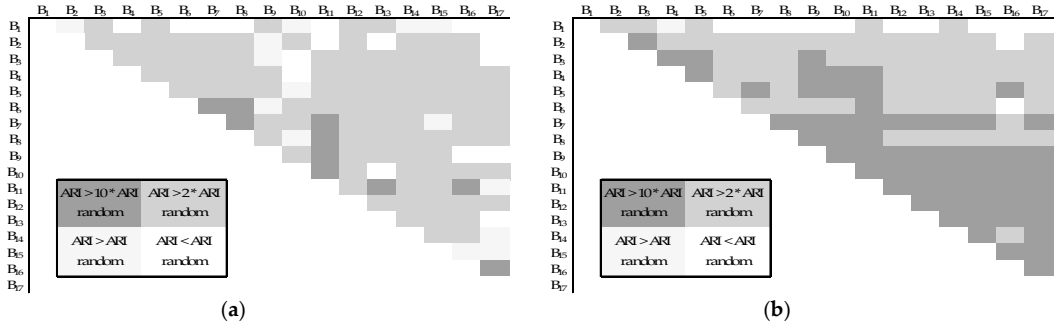


Figure A3. Detailed results for *minSeries* set at 1% and window length w set at 14 days.



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

Figure A4. ARI results for *minSeries* set at 1% and window length w set at 30 days (the short-term version is presented on the left part (**a**) while the long-term version is on the right part of the figure (**b**)).

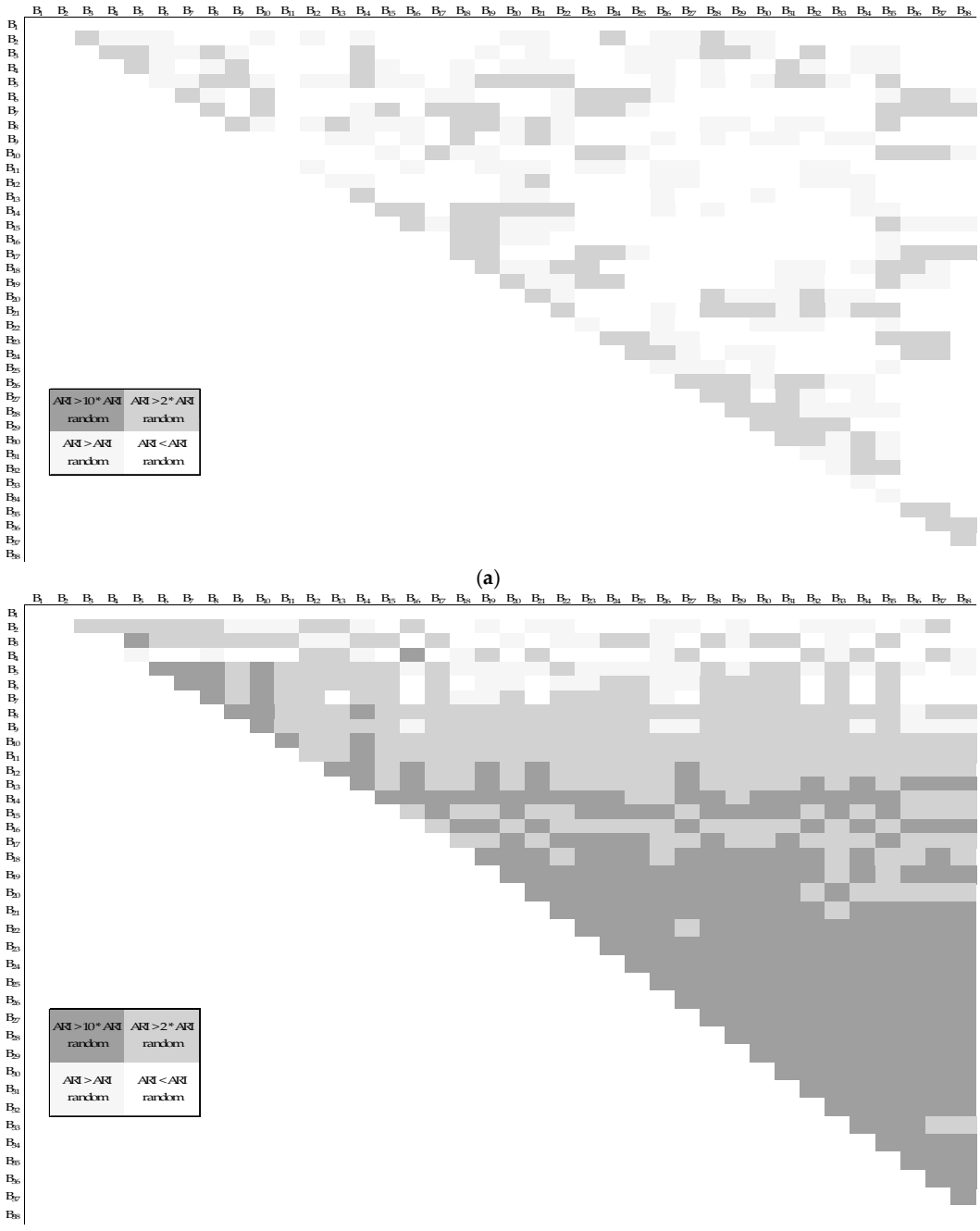
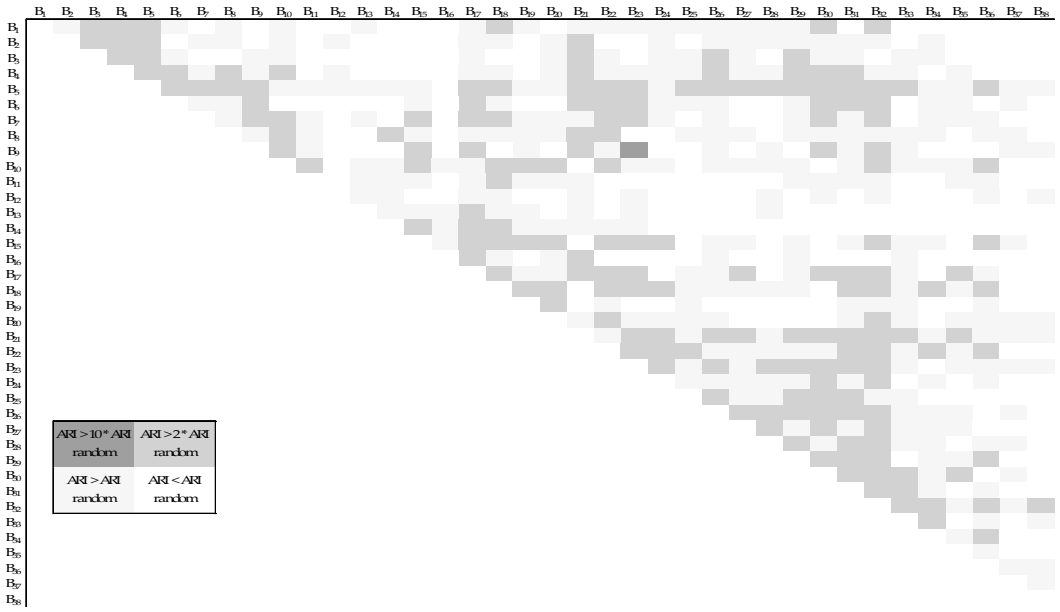
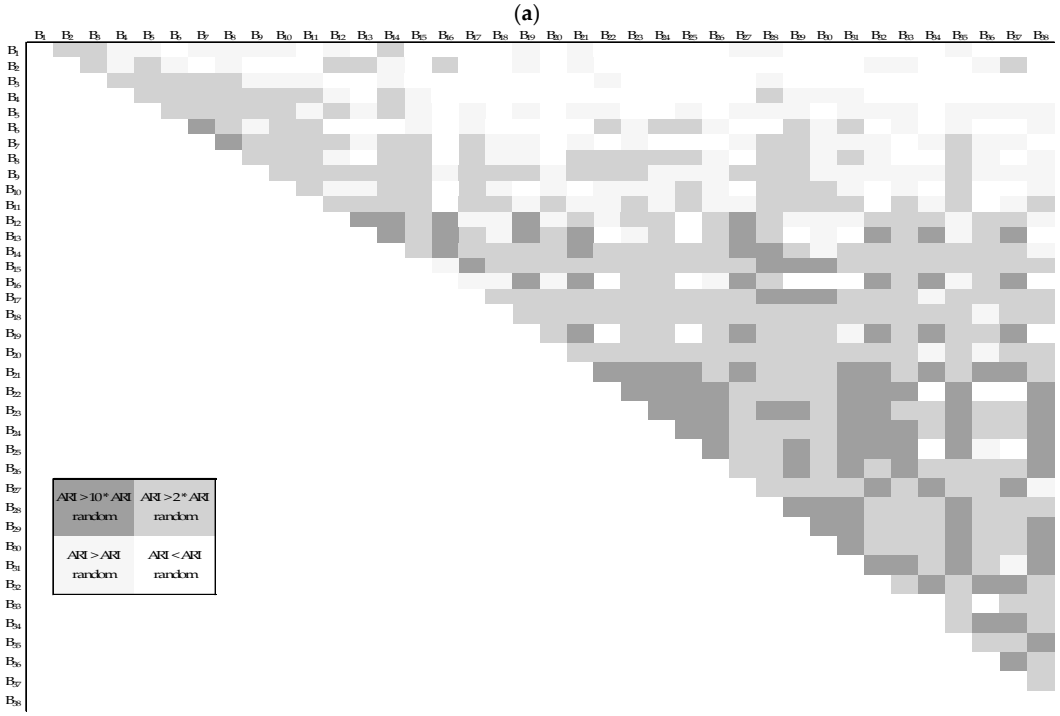


Figure A5. ARI results for *minSeries* set at 5% and window length w set at 14 days (the short-term version is presented on the upper part (**a**) while the long-term version is on the bottom part of the figure (**b**)).



(**a**)



(**b**)

Figure A6. ARI results for *minSeries* set at 1% and window length w set at 14 days (the short-term version is presented on the upper part (**a**) while the long-term version is on the bottom part of the figure (**b**)).

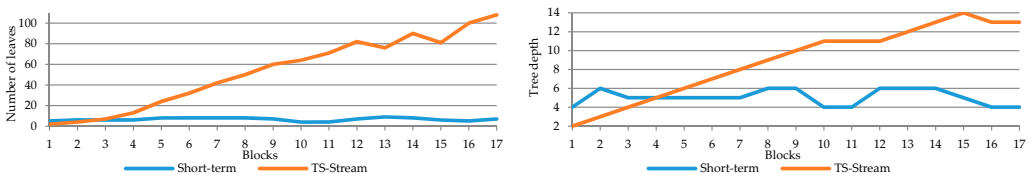


Figure A7. *Cont*.

Figure A7. Detailed results for *minSeries* set at 5% and window length w set at 30 days for basic TS-Stream algorithm.

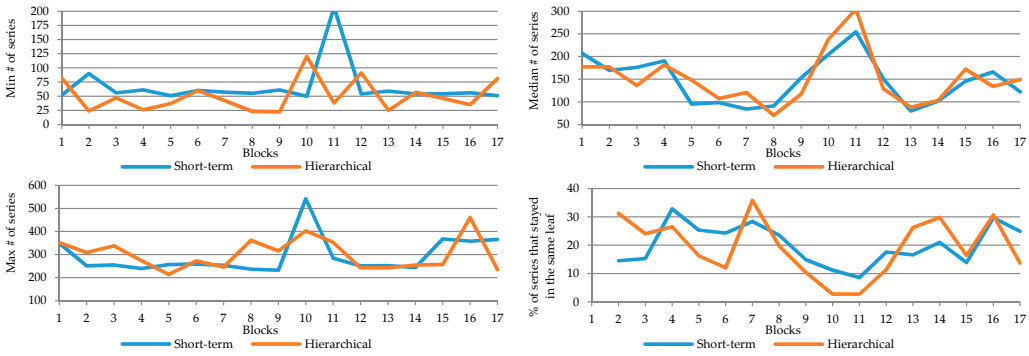


Figure A8. *Cont.*

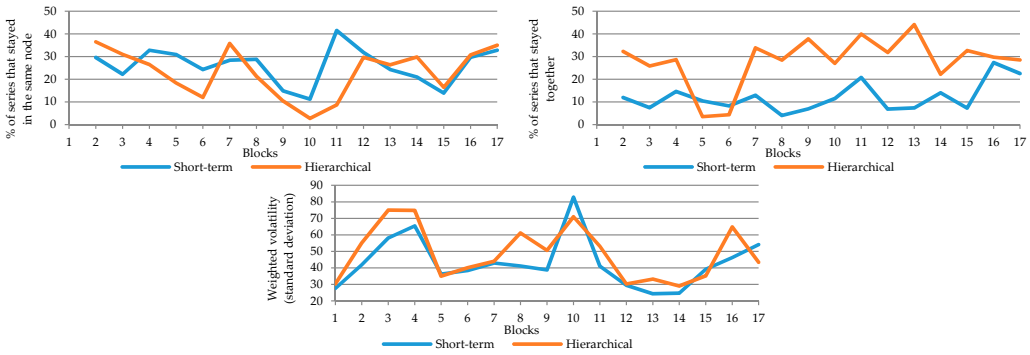


Figure A8. Detailed results for *minSeries* set at 5% and window length w set at 30 days for hierarchical clustering.

Appendix B

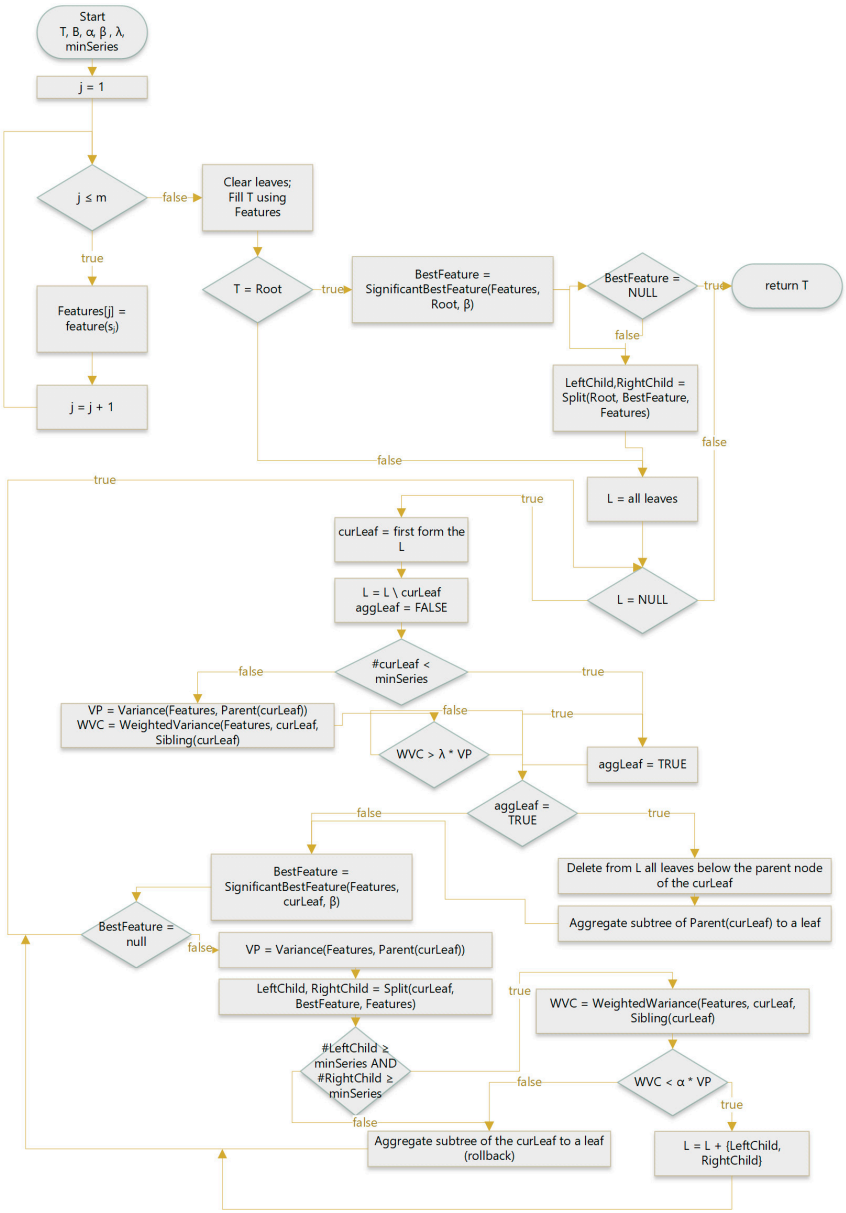


Figure A9. Flowchart for Algorithm 1 (for clarity, in some cases we have used abbreviations).

References

1. Gajowniczek, K.; Ząbkowski, T. Simulation Study on Clustering Approaches for Short-Term Electricity Forecasting. *Complexity* **2018**, *2018*, 1–21. [CrossRef]
2. Aletti, G.; Micheletti, A. A clustering algorithm for multivariate data streams with correlated components. *J. Big Data* **2017**, *4*, 48. [CrossRef]
3. Bones, C.C.; Romani, L.A.; de Sousa, E.P. Clustering Multivariate Climate Data Streams using Fractal Dimension. In Proceedings of the SBBD 2015, Rio de Janeiro, Brazil, 13–16 October 2015; pp. 41–52.
4. Pereira, C.M.M.; de Mello, R.F. TS-stream: Clustering time series on data streams. *J. Intell. Inf. Syst.* **2014**, *42*, 531–566. [CrossRef]
5. Gama, J.; Žliobaitė, I.; Bifet, A.; Pechenizkiy, M.; Bouchachia, A. A survey on concept drift adaptation. *ACM Comput. Surv.* **2014**, *46*, 1–37. [CrossRef]
6. Laurinec, P.; Lucká, M. Interpretable multiple data streams clustering with clipped streams representation for the improvement of electricity consumption forecasting. *Data Min. Knowl. Discov.* **2018**, *33*, 413–445. [CrossRef]
7. Laurinec, P.; Lucká, M. Clustering-based forecasting method for individual consumers electricity load using time series representations. *Open Comput. Sci.* **2018**, *8*, 38–50. [CrossRef]
8. Khan, I.; Huang, J.Z.; Ivanov, K. Incremental density-based ensemble clustering over evolving data streams. *Neurocomputing* **2016**, *191*, 34–43. [CrossRef]
9. Chicco, G.; Napoli, R.; Postolache, P.; Scutariu, M.; Toader, C. Customer Characterization Options for Improving the Tariff Offer. *IEEE Power Eng. Rev.* **2002**, *22*, 60. [CrossRef]
10. Nafkha, R.; Gajowniczek, K.; Ząbkowski, T. Do Customers Choose Proper Tariff? Empirical Analysis Based on Polish Data Using Unsupervised Techniques. *Energies* **2018**, *11*, 514. [CrossRef]
11. Hahsler, M.; Bolanos, M.; Forrest, J. Introduction to stream: An Extensible Framework for Data Stream Clustering Research with R. *J. Stat. Softw.* **2017**, *76*, 1–50. [CrossRef]
12. Bifet, A.; Read, J.; Holmes, G.; Pfahringer, B. Streaming Data Mining with Massive Online Analytics (MOA). *Data Mining in Time Series and Streaming Databases* **2018**, 1–25. [CrossRef]
13. Nelli, F. Machine Learning with scikit-learn. *Python Data Anal.* **2015**, 237–264. [CrossRef]
14. McLoughlin, F.; Duffy, A.; Conlon, M. Characterising domestic electricity consumption patterns by dwelling and occupant socio-economic variables: An Irish case study. *Energy Build.* **2012**, *48*, 240–248. [CrossRef]
15. Silva, J.A.; Faria, E.R.; Barros, R.C.; Hruschka, E.R.; Carvalho, A.C.P.L.F.; de Gama, J. Data stream clustering. *ACM Comput. Surv.* **2013**, *46*, 1–31. [CrossRef]
16. Aggarwal, C.C.; Yu, P.S.; Han, J.; Wang, J. A Framework for Clustering Evolving Data Streams. In Proceedings of the 2003 VLDB Conference, Berlin, Germany, 9–12 September 2003; pp. 81–92. [CrossRef]
17. Chen, Y.; Tu, L. Density-based clustering for real-time stream data. In Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining—KDD '07, San Jose, CA, USA, 12–15 August 2007. [CrossRef]
18. Hahsler, M.; Bolaos, M. Clustering Data Streams Based on Shared Density between Micro-Clusters. *IEEE Trans. Knowl. Data Eng.* **2016**, *28*, 1449–1461. [CrossRef]
19. Amini, A.; Saboohi, H.; Herawan, T.; Wah, T.Y. MuDi-Stream: A multi density clustering algorithm for evolving data stream. *J. Netw. Comput. Appl.* **2016**, *59*, 370–385. [CrossRef]
20. Chen, J.-Y.; He, H.-H. A fast density-based data stream clustering algorithm with cluster centers self-determined for mixed data. *Inf. Sci.* **2016**, *345*, 271–293. [CrossRef]
21. Dai, B.-R.; Huang, J.-W.; Yeh, M.-Y.; Chen, M.-S. Adaptive Clustering for Multiple Evolving Streams. *IEEE Trans. Knowl. Data Eng.* **2006**, *18*, 1166–1180. [CrossRef]
22. Chen, Y. Clustering Parallel Data Streams. *Data Min. Knowl. Discov. Real Life Appl.* **2009**. [CrossRef]
23. Beringer, J.; Hllermeier, E. Fuzzy Clustering of Parallel Data Streams. *Adv. Fuzzy Clust. Appl.* **2007**, 333–352. [CrossRef]
24. Rodrigues, P.P.; Gama, J.; Pedroso, J.P. Hierarchical Clustering of Time-Series Data Streams. *IEEE Trans. Knowl. Data Eng.* **2008**, *20*, 615–627. [CrossRef]
25. Chen, L.; Zou, L.-J.; Tu, L. A clustering algorithm for multiple data streams based on spectral component similarity. *Inf. Sci.* **2012**, *183*, 35–47. [CrossRef]

26. Vinh, N.X.; Epps, J. A Novel Approach for Automatic Number of Clusters Detection in Microarray Data Based on Consensus Clustering. In Proceedings of the 2009 Ninth IEEE International Conference on Bioinformatics and BioEngineering, Taichung, Taiwan, 22–24 June 2009. [CrossRef]
27. Katsev, S.; L'Heureux, I. Are Hurst exponents estimated from short or irregular time series meaningful? *Comput. Geosci.* **2003**, *29*, 1085–1089. [CrossRef]
28. Hurst, H.E. Methods of using long-term storage in reservoirs. *Proc. Inst. Civ. Eng.* **1956**, *5*, 519–543. [CrossRef]
29. Ersoy, O.K. A comparative review of real and complex Fourier-related transforms. *Proc. IEEE* **1994**, *82*, 429–447. [CrossRef]
30. Hssina, B.; Merbouha, A.; Ezzikouri, H.; Erritali, M. A comparative study of decision tree ID3 and C4.5. *Int. J. Adv. Comput. Sci. Appl.* **2014**, 4. [CrossRef]
31. Gajowniczek, K.; Orłowski, A.; Ząbkowski, T. Simulation Study on the Application of the Generalized Entropy Concept in Artificial Neural Networks. *Entropy* **2018**, *20*, 249. [CrossRef]
32. Rand, W.M. Objective criteria for the evaluation of clustering methods". Journal of the American Statistical Association. *Am. Stat. Assoc.* **1971**, *66*, 846–850. [CrossRef]
33. Rosenberg, A.; Hirschberg, J. V-measure: A conditional entropy-based external cluster evaluation measure. In Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), Prague, Czech Republic, 28–30 June 2007; pp. 410–420.
34. Gajowniczek, K.; Ząbkowski, T.; Sodenkamp, M. Revealing Household Characteristics from Electricity Meter Data with Grade Analysis and Machine Learning Algorithms. *Appl. Sci.* **2018**, *8*, 1654. [CrossRef]
35. Ramírez-Gallego, S.; Krawczyk, B.; García, S.; Woźniak, M.; Herrera, F. A survey on data preprocessing for data stream mining: Current status and future directions. *Neurocomputing* **2017**, *239*, 39–57. [CrossRef]