Hindawi
Complexity
Volume 2018, Article ID 3683969, 21 pages
https://doi.org/10.1155/2018/3683969

*Research Article*

# Simulation Study on Clustering Approaches for Short-Term Electricity Forecasting

**Krzysztof Gajowniczek and Tomasz Ząbkowski**

*Department of Informatics, Warsaw University of Life Sciences (SGGW), Warsaw, Poland*

Correspondence should be addressed to Krzysztof Gajowniczek; krzysztof_gajowniczek@sggw.pl

Received 22 December 2017; Revised 21 February 2018; Accepted 11 March 2018; Published 26 April 2018

Academic Editor: Tiago Pinto



Advanced metering infrastructures such as smart metering have begun to attract increasing attention; a considerable body of research is currently focusing on load profiling and forecasting at different scales on the grid. Electricity time series clustering is an effective tool for identifying useful information in various practical applications, including the forecasting of electricity usage, which is important for providing more data to smart meters. This paper presents a comprehensive study of clustering methods for residential electricity demand profiles and further applications focused on the creation of more accurate electricity forecasts for residential customers. The contributions of this paper are threefold: (1) using data from 46 homes in Austin, Texas, the similarity measures from different time series are analyzed; (2) the optimal number of clusters for representing residential electricity use profiles is determined; and (3) an extensive load forecasting study using different segmentation-enhanced forecasting algorithms is undertaken. Finally, from the operator's perspective, the implications of the results are discussed in terms of the use of clustering methods for grouping electrical load patterns.

## 1. Introduction

Throughout the EU, there is considerable interest in smart electricity networks, where increased control over electricity supply and consumption is achieved by investments and improvements in new technologies such as advanced metering infrastructure. Smart metering is part of this movement and is perceived as a necessary step in achieving the EU's energy policy goals by the year 2020 (i.e., cut greenhouse gas emissions by 20%, improve energy efficiency by 20%, and ensure that 20% of the EU's energy demand is supplied by renewable sources) [1].

Clustering analysis is an unsupervised learning technique that has been widely used to identify different energy consumption patterns, particularly among commercial customers, individual industries, or large aggregations of residential customers [2]. Recently, the fast-growing stream of meter data has motivated further research on the application of clustering techniques to individual residential customers [3]. By clustering time series of hourly load data, each customer can be represented by a number of load patterns, thus allowing variability information to be derived. Clustering can therefore serve as a valuable preprocessing step, providing fine-grained information on customer attributes and sources of variation for subsequent modeling and customer segmentation.

Many dissimilarity measures between time series have been proposed in the literature. They can be grouped into four categories [4, 5]:

(i) Shape: Minkowski distance, short time series distance, and dynamic time warping distance.

(ii) Editing: edit distance for real sequences and longest common subsequence distance.

(iii) Features: autocorrelation-based distances, short time series distance, Fourier coefficients-based distance, TQuest distance, and periodogram-based distances.

(iv) Structure: Maharaja distance and Piccolo distance.

This study examines the first three categories, as they can be used in the majority of cases. The fourth category requires

some a priori assumptions that affect the obtained results. For instance, they assume that the observed time series is the result of a certain parametric base model, mainly the autoregressive integrated moving-average (ARIMA) model. This implies the need to prioritize the relevant parameters of the model in advance.

Dissimilarity between time series can be also computed using information theory. For instance, Kullback–Leibler (KL) divergence is a measure of how one time series probability distribution diverges from a second (expected) probability distribution [6]. This measure is distribution-wise and asymmetric and thus does not qualify as a statistical metric of spread. In the simple case, a KL divergence of 0 indicates that we can expect similar, if not the same, behavior of two time series distributions, while a KL divergence of 1 indicates that the two time series distributions behave in such a different manner that the expectation given the first distribution approaches zero. The second example is mutual information (MI) of two time series which is a measure of the mutual dependence between two investigated time series [7]. Intuitively, mutual information measures the information that two time series share: it gives the idea of how one of these time series reduces uncertainty about the other. For example, if two time series are independent, then knowing the first time series does not give any information about the second time series and vice versa, so their mutual information is zero. At the other extreme, if the first time series is a deterministic function of the second one, then the mutual information is maximal.

As their input, most clustering algorithms take parameters such as the number of clusters, density of clusters, or, at least, the number of points in a cluster. Nonhierarchical procedures usually require the user to specify the number of clusters before any clustering is accomplished, whereas hierarchical methods routinely produce a series of solutions ranging from $q$ clusters to only a single cluster [8]. As such, the problems of determining a suitable number of clusters for a dataset and evaluating the clustering results have been considered by several researchers. The procedure of evaluating the results of a clustering algorithm is known as cluster validity analysis [9].

In general, there are three approaches for investigating the cluster validity. The first, based on external criteria, compares the cluster analysis results to some known results, such as externally provided class labels. The second approach, based on internal criteria, uses information obtained from within the clustering process to evaluate how well the results fit the data without reference to external information. The third approach is based on relative criteria and consists of evaluating a clustering structure by comparing it with other structures given by the same algorithm using different parameter values (e.g., the number of clusters). In this paper, we consider this third class of measures [10].

Based on the selected time series similarity measures and hierarchical clustering algorithms, this study developed forecasting models for the aggregate electricity demand of individual groups of households. The predictions given by these models were compared with the results of a base model built for all households (aggregate over 46 consumers for *WikiEnergy* data [11]). In particular, using smart metering data, we aim to answer the following research questions:

(1) To what extent is it possible to provide accurate 24-hour load forecasting for the group of households?

(2) Which of the proposed time series similarity measures and the measures determining the relevant number of clusters gives the greatest increase in forecast accuracy?

(3) What kind of forecasting methods and algorithms is appropriate to address highly volatile data?

The remainder of this paper is organized as follows. In Section 2, various time series similarity measures, grouped into three categories, are introduced. As different clustering algorithms usually lead to different numbers of clusters, Section 3 discusses several measures for determining the relevant number of clusters. In Section 4, based on *WikiEnergy* data gathered from 46 households, various numerical experiments regarding the clustering of these households are presented. Section 5 describes the methods used in our forecasting experiments, and Section 6 presents the results from a number of numerical experiments that provide 24-hour forecasts at different data aggregation levels. Finally, Section 7 concludes the paper.

## 2. Time Series Similarity Measures

The following subsections briefly describe three categories of time series similarity measures. In the remainder of this section, unless otherwise specified, $z_g = (z_1, z_2, \ldots, z_n)^T$ and $y_g = (y_1, y_2, \ldots, y_n)^T$ denote partial realizations from two real-valued processes $\{Z_g,\ g \in \mathbb{N}\}$ and $\{Y_g,\ g \in \mathbb{N}\}$, respectively. Note that serial realizations of the same length $n$ are initially assumed, although this limitation can be omitted in some cases.

*2.1. Measures Based on the Shape of Time Series.* A simple approach for measuring the similarity between $z_g$ and $y_g$ is to consider conventional metrics based on the closeness of their values at specific points in time. Some commonly used raw-values-based dissimilarity measures are introduced below.

*2.1.1. Minkowski Distance.* The Minkowski distance of order $p$, where $p$ is a positive integer, is also known as the $L_p$-norm distance [12]. This measure is typically used with $p = 2$, giving the Euclidean distance, and is very sensitive to signal transformations such as shifting or time scaling (stretching or shrinking of the time axis). The proximity notion relies on the closeness of the values observed at corresponding points in time, so that the observations are treated as if they were independent. In particular, $L_p$ is invariant to permutations over time [4, 5].

*2.1.2. Short Time Series Distance.* The short time series (STS) distance was introduced by Möller-Levet et al. [13] as a metric that adapts to the characteristics of irregularly sampled series [4, 5].

*2.1.3. Dynamic Time Warping Distance.* The goal of dynamic time warping (DTW) is to find patterns in time series [14]. The DTW distance determines a mapping between the series which minimizes a specific distance measure between the coupled observations. This allows similar shapes to be recognized, even in the presence of signal transformations such as shifting and/or scaling, and ignores the temporal structure of the values because the proximity is based on the differences, regardless of the behavior around these values [4, 5].

*2.2. Measures Based on Editing the Time Series.* The edit distance, which was initially developed to calculate the similarity between two sequences of strings, is based on the idea of counting the minimum number of edit operations (delete, insert, and replace) that are necessary to transform one sequence into the other. The problem of working with real numbers is that it is difficult to find exact matching points in two different sequences and, therefore, the edit distance is not directly applicable.

*2.2.1. Edit Distance for Real Sequences.* The distance between points in the time series is reduced to 0 or 1 [15]. If two points $z_g$ and $y_g$ are closer to each other in the absolute sense than some user-specified threshold $\varepsilon$, they are considered to be equal. On the contrary, if they are further apart, they are considered to be distinct and the distance between them is set to 1. As an additional property, the edit distance for real sequences (EDR) permits gaps or unmatched regions in the time series but penalizes them with a value equal to their length values [4, 5].

*2.2.2. Longest Common Subsequence Distance.* In this metric, the similarity between two time series is quantified in terms of the longest common subsequence (LCSS), with gaps or unmatched regions permitted [16]. As with EDR, the initial mapping between the series uses the Euclidean distance between two points, which is reduced to 0 or 1 depending on some threshold $\varepsilon$ values [4, 5].

*2.3. Measures Based on the Time Series Features.* Instead of using the raw data values in the series, this category of distance measures aims to extract a set of features from the time series and calculate the similarity between these features.

*2.3.1. Distance Based on the Cross-Correlation.* This distance is based on the cross-correlation between two time series. The maximum lag considered in the calculation should not exceed the length of the series values [4, 5].

*2.3.2. Autocorrelation-Based Distances.* Several researchers have considered measures based on the estimated autocorrelation functions [17]. These rely on the autocorrelation between two time series with a maximum lag of $L$ values [4, 5].

*2.3.3. Periodogram-Based Distances.* Most of the measures discussed so far operate in the same domain, that is, the time domain. However, the signal representation in the frequency domain provides a good alternative for measuring the similarity between time series. The key idea is to assess the similarity between the corresponding spectral representations of the time series values [4, 5, 18].

*2.3.4. Fourier Coefficients-Based Distances.* This measure is based on comparing the discrete Fourier transform coefficients of the series [19]. The value of each coefficient measures the contribution of its associated frequency to the series. Based on this, the inverse Fourier transform provides a means of representing the sequences as a combination of sinusoidal forms. Note that the Fourier coefficients are complex numbers that can be expressed as $z_{g_f} = a_f + b_f i$. In the case of real sequences such as time series, the discrete Fourier transform is symmetric, and therefore it is sufficient to study the first $n/2 + 1$ coefficients. Furthermore, it is commonly considered that most of the information is found within the first $n$ Fourier coefficients, where $t < n/2+1$. Based on this information, the distance between two time series is given by the Euclidean distance between the first $t$ coefficients [4, 5].

*2.3.5. TQuest Distance.* The fundamental idea of the TQuest distance [20] is to define a set of intervals in a time series in which the stochastic processes exceed a predetermined threshold $\tau$. The final distance between two time series is defined in terms of the similarity between sets of intervals based on this threshold value. Intuitively, two time intervals are said to be similar if they have similar start and end points. TQuest is independent of the size of the individual time series; this is important because time series exhibiting similar properties can be converted into intervals of different lengths. Another advantage is that this measure only takes into account the local similarity, with the remaining (continued) intervals not affecting the final result [4, 5].

## 3. Measures for Determining the Relevant Number of Clusters

Different clustering algorithms usually lead to different clusters of data; even for the same algorithm, the selection of different parameters or the order in which data objects are presented can greatly affect the final clustering partitions. Thus, effective evaluation standards and criteria are critically important to ensure confidence in the clustering results. At the same time, these assessments provide meaningful insights into how many clusters are hidden in the data. In most real-life clustering situations, users face the dilemma of selecting the number of clusters or partitions in the underlying data. As such, numerous indices for determining the number of clusters in a dataset have been proposed.

*3.1. CH Index.* The value of $q$ that maximizes CH($q$) specifies the number of clusters [21].

*3.2. C Index.* The minimum value of $C$ [21] is considered to be the relevant number of clusters in a given dataset.

*3.3. Duda Index.* Reference [21] proposed the ratio Je(2)/Je(1) as a criterion, where Je(2) is the sum of squared errors within clusters when the data are partitioned into two clusters and Je(1) is the squared error when only one cluster is present. The optimal number of clusters is that which gives the smallest value of this ratio.

*3.4. Ptbiserial Index.* This index [21] is simply a point-biserial correlation between the raw input dissimilarity matrix and a corresponding matrix consisting of 0s or 1s. A value of 0 is assigned if the two corresponding points are clustered together by the algorithm; a value of 1 is assigned otherwise. Given that larger positive values reflect a better fit between the data and the obtained partition, the maximum value of the index is used to select the optimal number of clusters in the dataset.

*3.5. DB Index.* This index [21] is a function of the ratio of within-cluster scatter to between-cluster separation. The $q$ value that minimizes the index is considered to be the optimal number of clusters in a given dataset.

*3.6. Frey Index.* The Frey index [21] can only be applied to hierarchical methods; it is the ratio of difference scores from two successive levels in the hierarchy. The numerator is the difference between the mean between-cluster distances from the two hierarchy levels (level $j$ and level $j + 1$), whereas the denominator is the difference between the mean within-cluster distances of levels $j$ and $j+1$.

*3.7. Hartigan Index.* The maximum difference between hierarchy levels is taken to indicate the correct number of clusters in the data [21].

*3.8. Ratkowsky Index.* Charrad et al. [21] proposed a criterion for determining the optimal number of clusters based on $\overline{S}/q^{0.5}$. The value of $\overline{S}$ is the average of the ratios of $(\mathrm{BGSS}_j/\mathrm{TSS}_j)$, where BGSS is the sum of squares between the clusters (groups) for each variable and TSS is the total sum of squares for each variable. The optimal number of clusters is the value of $q$ that maximizes $\overline{S}/q^{0.5}$.

*3.9. Ball Index.* Ball and Hall [21] proposed an index based on the average distance between data items and their respective cluster centroids. The largest difference between levels is used to indicate the optimal solution.

*3.10. McClain Index.* The McClain and Rao index [21] is the ratio of the average within-cluster distance divided by the number of within-cluster distances to the average between-cluster distance divided by the number of cluster distances. The minimum value of this index indicates the optimal number of clusters.

*3.11. KL Index.* The value of $q$ that maximizes KL($q$) [21] specifies the optimal number of clusters.

*3.12. Silhouette Index.* The maximum value of this index is used to determine the optimal number of clusters in the data [21].

*3.13. Dunn Index.* The Dunn index [21] is the ratio between the minimal intercluster distance and the maximal intracluster distance. If the dataset contains compact and well-separated clusters, the diameter of the clusters is expected to be small and the distance between the clusters is expected to be large. Thus, the Dunn index should be maximized.

*3.14. SD Index.* The SD validity index is based on the concepts of average scattering for clusters and total separation between clusters [21]. The number of clusters $q$ that minimizes this index gives the optimal value for the number of clusters in the dataset.

## 4. Splitting Households into Clusters

*4.1. Data Characteristics.* Numerical analyses were performed using data from 46 households taken from *WikiEnergy*. The *WikiEnergy* dataset, constructed by Pecan Street Inc., is a large database of consumer energy information. This database is highly granular, including usage measurements collected from up to 24 circuits within the home. The households considered in the analysis are located in Austin, Texas, USA. We extracted 14 months of data (March 2013–April 2014) from 46 households in nearby neighborhoods at a granularity level of 1 hour. Thus, it was possible to aggregate the hourly demand values and divide the consumers into homogeneous groups [1]. From the aggregated data (Figure 1), we could see that the highest electricity consumption takes place in summer, between June and August, most likely due to air conditioning.

To analyze the volatility of the load, we prepared the box and whisker plots, for each of the 24 hours over the whole year, for two customers: one with quite stable load profile and the second with highly volatile characteristics (see Figures 2 and 3 for details). The whiskers show the minimum and maximum value in a given hour and the box encloses 50% of the total data (top edge represents the 75th quartile and bottom edge the 25th quartile, and the line in the middle is the median). For instance, the household, as shown in Figure 2, on average consumes 1.5 kWh in each hour while the volatility is rather low. The other household, as shown in Figure 3, can be characterized as the one using, on average, only 0.2 kWh in each hour; however, the volatility of the load is very high, regardless of the hour.

It should be noted that the source dataset was of relatively high quality. The time series in the dataset had no missing values. In some time series, few outliers were observed; however, they were left in the dataset. Please refer to Table 1 for the descriptive statistics of the electric load observed at each of the households. In the preprocessing step, only normalization of the data was applied. Additionally, only few features describing particular households were available (i.e., structural and building specific data). Those were building type (apartment, single-family home, and town home),

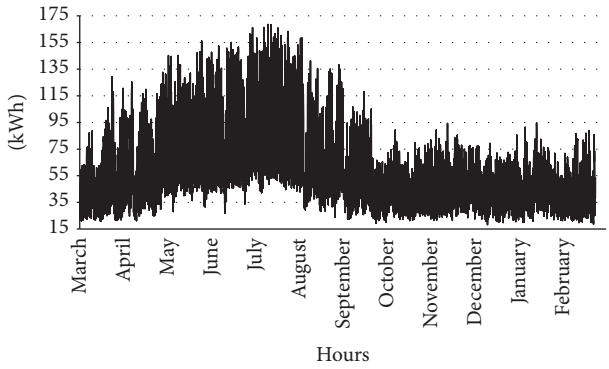


Figure 1: Hourly load data for 46 customers from 1 March 2013 to 28 February 2014.

construction year, total square footage, and incentive program due to the installed photovoltaic systems [22].

The dataset was split into training, validation, and testing samples. The training sample consisted of data from 365 days (8760 observations) between April 5, 2013, and February 28, 2014; the validation sample consisted of 28 days (672 observations) between March 1, 2014, and March 28, 2014; and the testing sample consisted of 14 days (336 observations) between March 29, 2014, and April 11, 2014 [1].

*4.2. Determining the Similarity between Time Series.* The starting point for the process of partitioning households was the selection of training samples for the period considered in determining the similarity between the electricity consumption time series.

In the second step, using the *R-CRAN* software, matrixes of similarity measures were calculated between all 46 time series. All measures were implemented in the following libraries:

(i) *TSdist* [5]: the *tsDatabaseDistances* function was used with arguments implementing the following similarity measures:

(a) *ccor*: distance based on the cross-correlation.
(b) *edr*: EDR with a threshold value of $\varepsilon = 0.2$.
(c) *dtw*: DTW distance.
(d) *euclidean*: $L_p$ distance.
(e) *fourier*: Fourier coefficients-based distance.
(f) *lcss*: LCSS distance with a threshold value of $\varepsilon = 0.2$.
(g) *sts*: STS distance.
(h) *tquest*: TQuest distance with a threshold value of $\tau = 1$.

(ii) *TSclust* [4]: the *diss* function was used with arguments implementing the following similarity measures:

(a) *ACF*: autocorrelation-based distances with a maximum delay of $L = 50$ and geometrically decaying weights with $p = 0.05$.

Table 1: Descriptive statistics for the electric load observed at each of the households.

| Household number | Missing | Min | Max | Mean | CV |
|---|---|---|---|---|---|
| 1 | 0 | 0.05 | 7.36 | 0.65 | 1.24 |
| 2 | 0 | 0.19 | 7.41 | 1.29 | 0.86 |
| 3 | 0 | 0.01 | 13.88 | 3.67 | 0.65 |
| 4 | 0 | 0.00 | 7.89 | 1.62 | 0.65 |
| 5 | 0 | 0.39 | 12.62 | 2.42 | 0.72 |
| 6 | 0 | 0.20 | 11.40 | 2.07 | 0.86 |
| 7 | 0 | 0.19 | 11.10 | 1.42 | 1.14 |
| 8 | 0 | 0.02 | 5.36 | 0.48 | 1.38 |
| 9 | 0 | 0.27 | 5.85 | 1.34 | 0.63 |
| 10 | 0 | 0.00 | 9.20 | 1.06 | 1.27 |
| 11 | 0 | 0.36 | 10.55 | 1.77 | 0.78 |
| 12 | 0 | 0.13 | 9.16 | 1.06 | 1.15 |
| 13 | 0 | 0.08 | 6.27 | 0.56 | 1.08 |
| 14 | 0 | 0.16 | 8.13 | 1.03 | 0.99 |
| 15 | 0 | 0.24 | 10.10 | 1.41 | 0.73 |
| 16 | 0 | 0.00 | 9.53 | 1.25 | 1.15 |
| 17 | 0 | 0.19 | 5.44 | 0.94 | 1,04 |
| 18 | 0 | 0.27 | 7,80 | 1,33 | 0,82 |
| 19 | 0 | 0.00 | 10.04 | 1.59 | 0.98 |
| 20 | 0 | 0.31 | 7.33 | 1.35 | 0.77 |
| 21 | 0 | 0.65 | 11.21 | 2.69 | 0.62 |
| 22 | 0 | 0.11 | 4.05 | 0.50 | 0.65 |
| 23 | 0 | 0.66 | 6.72 | 1.50 | 0.53 |
| 24 | 0 | 0.29 | 8.06 | 1.09 | 0.92 |
| 25 | 0 | 0.08 | 4.71 | 0.35 | 1.32 |
| 26 | 0 | 0.15 | 7.02 | 1.11 | 0.97 |
| 27 | 0 | 0.08 | 5.70 | 0.51 | 0.98 |
| 28 | 0 | 0.00 | 7.96 | 1.38 | 0.83 |
| 29 | 0 | 0.18 | 10.55 | 1.27 | 1.09 |
| 30 | 0 | 0.00 | 14.46 | 3.01 | 0.77 |
| 31 | 0 | 0.08 | 7.88 | 0.76 | 1.17 |
| 32 | 0 | 0.25 | 7.23 | 1.06 | 0.85 |
| 33 | 0 | 0.15 | 7.15 | 1.12 | 1.11 |
| 34 | 0 | 0.26 | 10.67 | 1.27 | 1.01 |
| 35 | 0 | 0.20 | 8.14 | 1.64 | 0.85 |
| 36 | 0 | 0.06 | 5.10 | 0.48 | 0.90 |
| 37 | 0 | 0.21 | 10.15 | 1.33 | 0.92 |
| 38 | 0 | 0.00 | 3.81 | 0.56 | 0.67 |
| 39 | 0 | 0.00 | 11.45 | 1.40 | 1.16 |
| 40 | 0 | 0,00 | 8,49 | 1,03 | 1,08 |
| 41 | 0 | 0,15 | 6,58 | 0,92 | 0,92 |
| 42 | 0 | 0.10 | 3.90 | 0.43 | 0.89 |
| 43 | 0 | 0.00 | 6.97 | 1.06 | 0.94 |
| 44 | 0 | 0.06 | 4.31 | 0.31 | 1.17 |
| 45 | 0 | 0.41 | 13.77 | 2.45 | 0.84 |
| 46 | 0 | 0.10 | 5.68 | 0.75 | 0.91 |

(b) *INT.PER*: periodogram-based distances in non-standardized version.

The households were divided into clusters using the Ward method, contained in the *hclust* function. Each of the 10

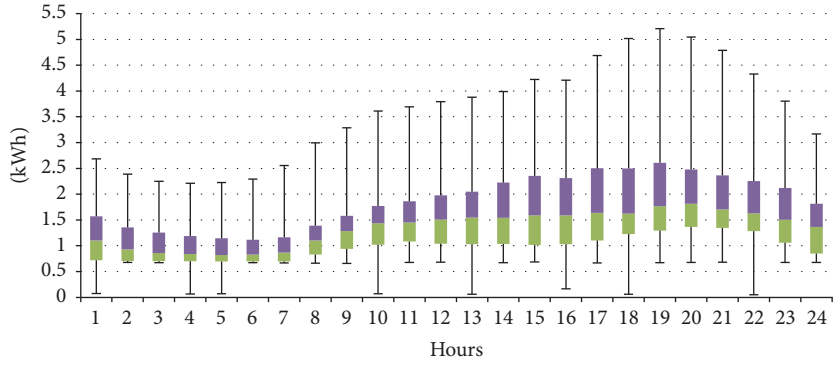


Figure 2: Customer with the least volatile consumption (in kWh) in the analyzed period (1 March 2013 to 28 February 2014).

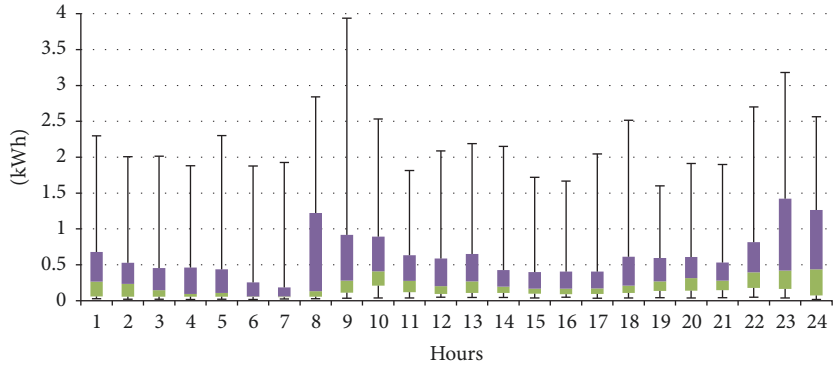


Figure 3: Customer with the most volatile consumption (in kWh) in the analyzed period (1 March 2013 to 28 February 2014).

resulting dendrograms was then divided into the following groups by the *cutree* function:

(i) Partition 1 containing 2 clusters.
(ii) Partition 2 containing 3 clusters.
(iii) Partition 3 containing 4 clusters.
(iv) Partition 4 containing 5 clusters.
(v) Partition 5 containing 6 clusters.
(vi) Partition 6 containing 7 clusters.
(vii) Partition 7 containing 8 clusters.
(viii) Partition 8 containing 9 clusters.

The size of each cluster according to the type of partition and the similarity of hourly time series is listed in Table 2. For the same type of division, the number of households included in the individual clusters varies depending on the similarity measure. The quality of the proposed similarity measures when splitting the data into two clusters is shown in Figure 4.

The most unbalanced group sizes are given by the distance based on cross-correlation (one very large cluster, the others very small), STS distance (two large clusters, the others very small), and TQuest distance (one very large cluster). The most balanced group sizes are given by autocorrelation-based distances (the smallest cluster contains three households), EDR (only in the seventh division do groups with fewer than three households appear), and periodogram-based distances (only in the last split do groups with fewer than three households appear). Other similarity measures generate clusters with a similar number of customers, but small clusters are present at some levels of division.

To determine the similarity between the results of time series clustering (Table 3), Baker's correlation coefficient [23] (function *cor_bakers_gamma*) was implemented in the *dendextend* library. This coefficient measures the similarity between two dendrograms, that is, the results of a hierarchical clustering, by calculating the highest possible value of $q$ (number of groups when cutting a tree) for which two time series still belong to the same subtree. It is known that there are exactly $\binom{46}{2} = 1035$ combinations of such pairs of time series, based on hierarchical clustering. The same operation is then performed on the basis of the second dendrogram. In the next step, these two sets of values (for each dendrogram) are paired according to the order of comparing elements. Eventually, the similarity of the clustering results is assessed on the basis of Spearman's correlation coefficient.

Based on the results presented in Tables 2 and 3, it can be observed that similar clustering results are related to the DTW distance *(dtw)*, $L_p$ distance *(euclidean)*, Fourier coefficients-based distance *(fourier)*, and periodogram-based distances *(int.per)* (denoted by italic font in Table 3). Distances carrying the most dissimilar dendrograms are the autocorrelation-based distances and the longest common subsequence distance.

Table 2: Size of each cluster by the type of partition and time series similarity measure.

| Partition number | Autocorrelation-based distances | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 35 | 11 | | | | | | | |
| 2 | 26 | 11 | 9 | | | | | | |
| 3 | 23 | 11 | 3 | 9 | | | | | |
| 4 | 7 | 16 | 11 | 3 | 9 | | | | |
| 5 | 7 | 16 | 4 | 7 | 3 | 9 | | | |
| 6 | 7 | 11 | 4 | 5 | 7 | 3 | 9 | | |
| 7 | 7 | 11 | 4 | 5 | 7 | 3 | 4 | 5 | |
| 8 | 3 | 4 | 11 | 4 | 5 | 7 | 3 | 4 | 5 |

| Partition number | Fourier coefficients-based distance | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 40 | 6 | | | | | | | |
| 2 | 30 | 6 | 10 | | | | | | |
| 3 | 30 | 2 | 4 | 10 | | | | | |
| 4 | 30 | 1 | 4 | 10 | 1 | | | | |
| 5 | 21 | 1 | 9 | 4 | 10 | 1 | | | |
| 6 | 21 | 1 | 8 | 4 | 10 | 1 | 1 | | |
| 7 | 13 | 1 | 8 | 8 | 4 | 10 | 1 | 1 | |
| 8 | 13 | 1 | 8 | 8 | 3 | 1 | 10 | 1 | 1 |

| Partition number | Distance based on cross-correlation | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 40 | 6 | | | | | | | |
| 2 | 40 | 5 | 1 | | | | | | |
| 3 | 40 | 1 | 4 | 1 | | | | | |
| 4 | 39 | 1 | 4 | 1 | 1 | | | | |
| 5 | 39 | 1 | 2 | 1 | 2 | 1 | | | |
| 6 | 36 | 3 | 1 | 2 | 1 | 2 | 1 | | |
| 7 | 35 | 3 | 1 | 2 | 1 | 2 | 1 | 1 | |
| 8 | 35 | 3 | 1 | 1 | 1 | 2 | 1 | 1 | 1 |

| Partition number | Periodogram-based distances | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 37 | 9 | | | | | | | |
| 2 | 26 | 9 | 11 | | | | | | |
| 3 | 26 | 3 | 6 | 11 | | | | | |
| 4 | 10 | 3 | 16 | 6 | 11 | | | | |
| 5 | 10 | 3 | 11 | 6 | 11 | 5 | | | |
| 6 | 5 | 3 | 11 | 6 | 5 | 11 | 5 | | |
| 7 | 5 | 3 | 11 | 3 | 5 | 3 | 11 | 5 | |
| 8 | 5 | 2 | 11 | 3 | 5 | 3 | 1 | 11 | 5 |

| Partition number | DTW distance | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 40 | 6 | | | | | | | |
| 2 | 30 | 6 | 10 | | | | | | |
| 3 | 24 | 6 | 6 | 10 | | | | | |
| 4 | 24 | 2 | 6 | 4 | 10 | | | | |
| 5 | 24 | 1 | 6 | 4 | 10 | 1 | | | |
| 6 | 15 | 1 | 6 | 9 | 4 | 10 | 1 | | |
| 7 | 15 | 1 | 6 | 9 | 1 | 3 | 10 | 1 | |
| 8 | 6 | 1 | 9 | 6 | 9 | 1 | 3 | 10 | 1 |

| Partition number | LCSS distance | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 40 | 6 | | | | | | | |
| 2 | 38 | 2 | 6 | | | | | | |
| 3 | 9 | 29 | 2 | 6 | | | | | |
| 4 | 9 | 12 | 2 | 17 | 6 | | | | |
| 5 | 4 | 12 | 2 | 17 | 6 | 5 | | | |
| 6 | 4 | 12 | 2 | 16 | 6 | 5 | 1 | | |
| 7 | 4 | 7 | 2 | 5 | 16 | 6 | 5 | 1 | |
| 8 | 4 | 6 | 2 | 5 | 16 | 6 | 5 | 1 | 1 |

| Partition number | EDR distance | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 24 | 22 | | | | | | | |
| 2 | 24 | 7 | 15 | | | | | | |
| 3 | 18 | 7 | 15 | 6 | | | | | |
| 4 | 14 | 7 | 15 | 6 | 4 | | | | |
| 5 | 14 | 3 | 15 | 4 | 6 | 4 | | | |
| 6 | 12 | 3 | 15 | 4 | 2 | 6 | 4 | | |
| 7 | 12 | 2 | 15 | 4 | 2 | 6 | 1 | 4 | |
| 8 | 12 | 2 | 7 | 4 | 8 | 2 | 6 | 1 | 4 |

| Partition number | STS distance | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 25 | 21 | | | | | | | |
| 2 | 24 | 21 | 1 | | | | | | |
| 3 | 23 | 1 | 21 | 1 | | | | | |
| 4 | 22 | 1 | 21 | 1 | 1 | | | | |
| 5 | 21 | 1 | 21 | 1 | 1 | 1 | | | |
| 6 | 19 | 1 | 21 | 2 | 1 | 1 | 1 | | |
| 7 | 19 | 1 | 21 | 1 | 1 | 1 | 1 | 1 | |
| 8 | 18 | 1 | 21 | 1 | 1 | 1 | 1 | 1 | 1 |

| Partition number | $L_p$ distance | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 40 | 6 | | | | | | | |
| 2 | 30 | 6 | 10 | | | | | | |
| 3 | 30 | 2 | 4 | 10 | | | | | |
| 4 | 30 | 1 | 4 | 10 | 1 | | | | |
| 5 | 18 | 1 | 12 | 4 | 10 | 1 | | | |
| 6 | 18 | 1 | 11 | 4 | 10 | 1 | 1 | | |
| 7 | 18 | 1 | 10 | 1 | 4 | 10 | 1 | 1 | |
| 8 | 18 | 1 | 6 | 4 | 1 | 4 | 10 | 1 | 1 |

| Partition number | TQuest distance | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 42 | 4 | | | | | | | |
| 2 | 42 | 3 | 1 | | | | | | |
| 3 | 41 | 3 | 1 | 1 | | | | | |
| 4 | 41 | 1 | 2 | 1 | 1 | | | | |
| 5 | 39 | 1 | 2 | 1 | 2 | 1 | | | |
| 6 | 35 | 4 | 1 | 2 | 1 | 2 | 1 | | |
| 7 | 35 | 3 | 1 | 2 | 1 | 2 | 1 | 1 | |
| 8 | 35 | 3 | 1 | 2 | 1 | 1 | 1 | 1 | 1 |

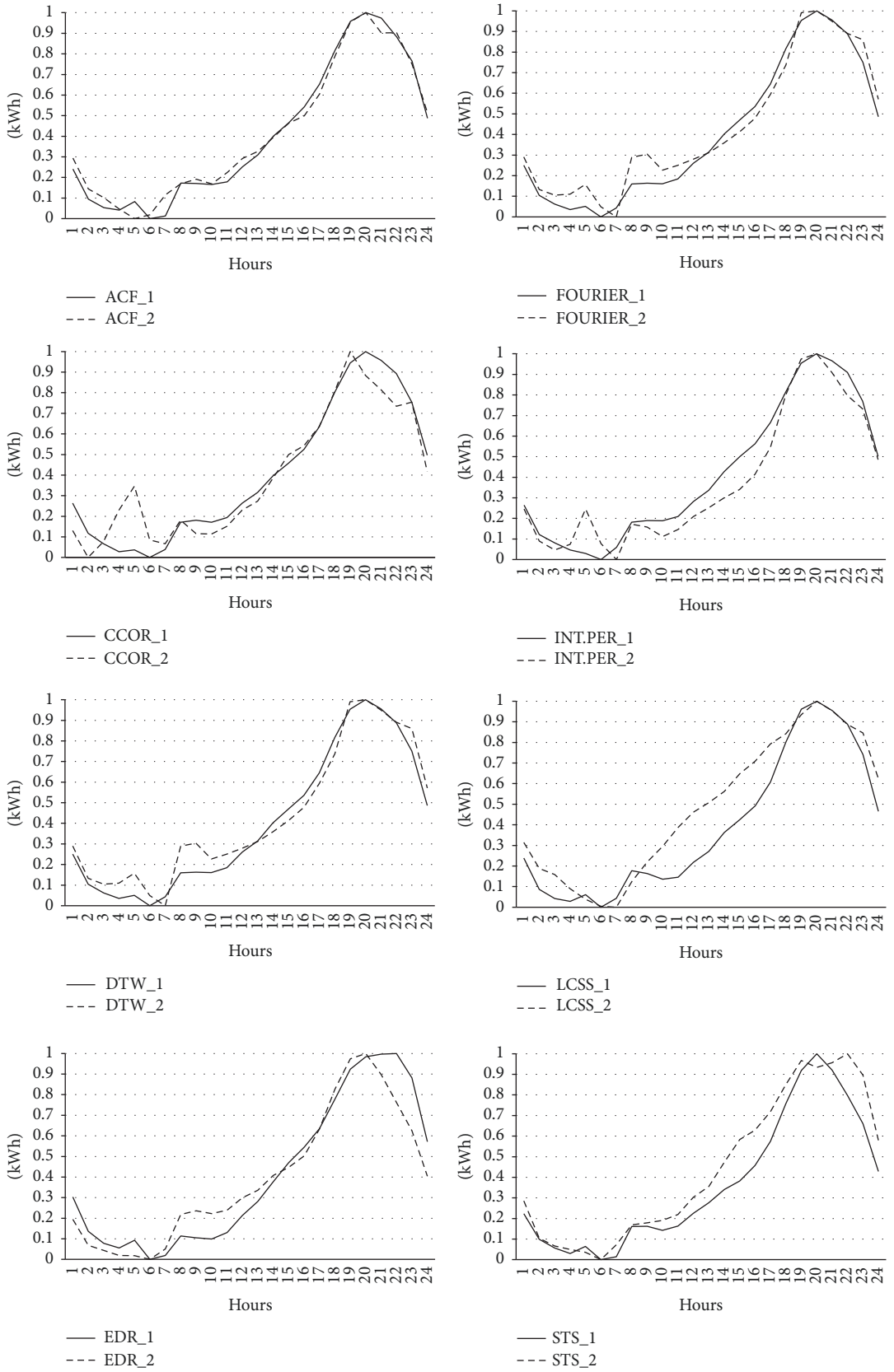


Figure 4: Continued.

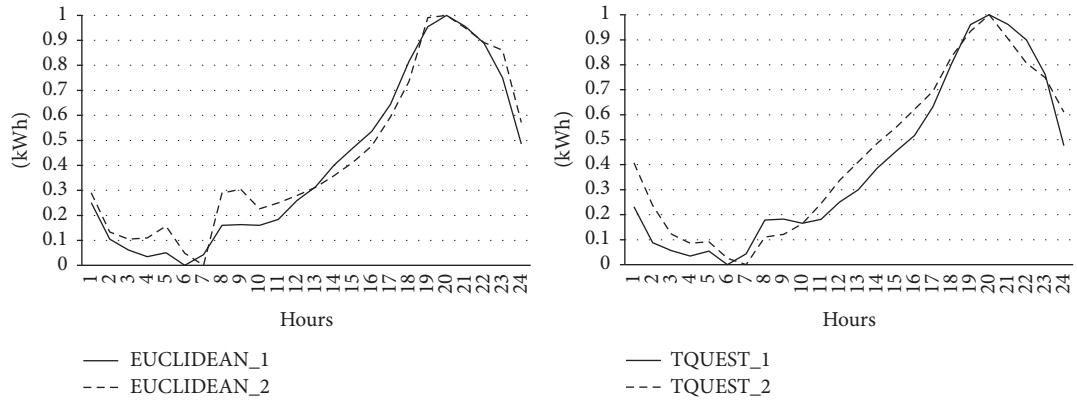


Figure 4: Normalized average hourly load profiles for partition 1 containing 2 clusters.

Table 3: Baker's correlation coefficient between clustering results on the basis of different measures of time series similarity.

| | acf | ccor | dtw | edr | euclidean | fourier | int.per | lcss | sts | tquest |
|---|---|---|---|---|---|---|---|---|---|---|
| acf | *1.000* | | | | | | | | | |
| ccor | 0.038 | *1.000* | | | | | | | | |
| dtw | 0.150 | 0.099 | *1.000* | | | | | | | |
| edr | 0.066 | 0.044 | 0.315 | *1.000* | | | | | | |
| euclidean | 0.132 | 0.121 | *0.948* | 0.292 | *1.000* | | | | | |
| fourier | 0.158 | 0.116 | *0.951* | 0.337 | *0.965* | *1.000* | | | | |
| int.per | 0.050 | 0.110 | *0.667* | 0.232 | *0.715* | *0.699* | *1.000* | | | |
| lcss | 0.024 | −0.053 | 0.001 | −0.039 | −0.003 | 0.003 | −0.060 | *1.000* | | |
| sts | 0.044 | 0 | 0.267 | 0.114 | 0.236 | 0.227 | 0.334 | −0.043 | *1.000* | |
| tquest | −0.021 | 0.350 | 0.148 | 0.084 | 0.174 | 0.139 | 0.135 | −0.007 | −0.024 | *1.000* |

*4.3. Determining the Relevant Number of Clusters.* Using the *NbClust* library [21], the relevant number of clusters for all 46 time series was determined. The input arguments of this package are the real-time values of the time series and a similarity array. The optimum number of clusters was determined on the basis of the eight different partitions defined earlier (Table 2). The input parameters for the *NbClust* function were as follows:

(i) *ch*: Calinski and Harabasz index.
(ii) *duda*: Duda index.
(iii) *cindex*: *C* index.
(iv) *ptbiserial*: Ptbiserial index.
(v) *db*: DB index.
(vi) *frey*: Frey index.
(vii) *hartigan*: Hartigan index.
(viii) *ratkowsky*: Ratkowsky index.
(ix) *ball*: Ball index.
(x) *mcclain*: McClain index.
(xi) *kl*: KL index.
(xii) *silhouette*: Silhouette index.
(xiii) *dunn*: Dunn index.
(xiv) *sdindex*: SD index.

The aggregated results for the relevant number of clusters for hourly electricity demand are presented in Table 4. Using a simple majority vote for the relevant number of clusters (partitions in italic font denote the greatest number of measures), it was observed that most measures of time series similarity would split the households into two groups. The LCSS distance gives a very different result, and the EDR and TQuest distances give two equally applicable cluster numbers.

Additionally, it can be realized that Ball's, Frey's, and McClain's measures tend to indicate relatively few clusters. For the remaining measures, some subgroups indicate a similar number of clusters for particular time series similarity measures [24]. For example, Silhouette's, Ptbiserial's, Davies and Bouldin's, Ratkowski's, Dunn's, and SD's measures indicate almost the maximum number of clusters for the LCSS and STS distances, whereas they indicate the minimum number of clusters for DTW, EDR, and the periodogram-based distances.

## 5. An Approach to Forecasting

*5.1. Short-Term Load Forecasting.* Recently, with advances in smart metering, there has been a lot of interest in residential power load forecasting and application of analytical techniques to load forecasting on the individual household level [1, 25–27]. However, such a level of granularity is challenging due to the extreme load volatility which is

Table 4: Voting results for the relevant number of clusters based on the partition type and the time series similarity measure.

| Measure distance | Number of clusters | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Autocorrelation-based distances | *5* | 2 | 3 | 1 | 0 | 0 | 0 | 2 |
| Distance based on cross-correlation | *4* | 1 | 1 | 1 | 2 | 0 | 1 | 3 |
| DTW distance | *7* | 5 | 1 | 0 | 1 | 0 | 0 | 0 |
| EDR distance | *6* | *6* | 0 | 0 | 0 | 1 | 0 | 0 |
| *Lp* distance | *8* | 2 | 2 | 2 | 0 | 0 | 0 | 0 |
| Periodogram-based distances | *7* | 3 | 0 | 0 | 0 | 0 | 2 | 2 |
| Fourier coefficients-based distance | *8* | 3 | 1 | 2 | 0 | 0 | 0 | 0 |
| LCSS distance | 2 | 1 | 0 | 3 | 1 | 0 | 1 | *5* |
| STS distance | *5* | 1 | 0 | 1 | 1 | 0 | 1 | 4 |
| TQuest distance | 2 | 2 | 1 | 1 | *3* | *3* | 0 | 1 |

the result of many different dynamic processes observed at individual households, including behavioral and sociodemographic components. Aggregation of individual and to some extent stochastic components reduces the inherent variability of electricity usage resulting in smoother load shapes, and, as a result, the forecasting applied to the higher aggregation levels (power stations or regions) can be achieved with quite low errors.

Different methods have been proposed for forecasting the electric load demand in the last decades. Several modeling techniques are typically used for energy load forecasting. These techniques can be classified into nine categories [28]: (1) multiple regression, (2) exponential smoothing, (3) iterative reweighted least squares, (4) adaptive load forecasting, (5) stochastic time series, (6) ARMAX models based on genetic algorithms, (7) fuzzy logic, (8) artificial neural networks, and (9) expert systems. However, the list is not closed and new techniques are still being adopted and tested for the purpose of accurate electricity load forecasting, including support vector machines or random forests, just to name a few.

Based on literature review, one can conclude that time series analyses are not effective in highly volatile data [29], and therefore time series methods such as regression models, ARIMA models, GARCH, and hybrid models such as the combination of ARIMA and GARCH using wavelet transform are not considered for short-term forecasting when dealing with volatile data [25, 27]. Rather than these, the machine learning techniques like artificial neural networks, support vector machines, or random forests are recently applied in this area with positive outcome [30–33].

Therefore, for the forecasting experiments, we focused on application of machine learning techniques including artificial neural networks, support vector machines, random regression forests, regression trees, and $k$-NN regression, and as benchmark models, we have selected ARIMA models and some simple techniques like stepwise regression, naive forecast, and random forecasts.

*5.2. Accuracy Measures.* To assess the forecasting performance of the model, three measures were used: precision, resistant mean absolute percentage error (MAPE), and accuracy [34]. Precision is defined as how well the model is able to forecast the actual load. To measure the precision, the mean squared error (MSE) was used:

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(L_i - P_i\right)^2, \tag{1}$$

where $L_i$ and $P_i$ denote the observed and forecasted load in hour $i$, respectively [1].

The MAPE satisfies the criteria of reliability, ease of interpretation, and clarity of presentation. However, it does not meet the validity criterion, because the distribution of the absolute percentage errors is usually skewed to the right because of the presence of outliers. In these cases, MAPE can be overinfluenced by some very bad instances, which disrupt otherwise good forecasts [34]. Therefore, we propose the alternative measure of resistant MAPE (r-MAPE) based on the calculation of Huber's M-estimator, which helps to overcome the aforementioned limitation [35].

The M-estimator for the location parameter $\mu$ using the maximum likelihood (ML) estimator is defined as a solution $\theta$ to

$$\min_{\theta} \quad \sum_{i=1}^{n}\rho\left(\frac{\left|\left(L_i - P_i\right)/L_i\right| - \theta}{\sigma}\right) \times 100\%, \tag{2}$$

or

$$\sum_{i=1}^{n}\varphi\left(\frac{\left|\left(L_i - P_i\right)/L_i\right| - \theta}{\sigma}\right) \times 100\% = 0, \tag{3}$$

where $\varphi = \rho'$ and $\sigma$ is the scale parameter. For a given positive constant $k$, Huber's estimator is defined by the following function $\varphi$ in (3):

$$\varphi(k) = \begin{cases} k & x > k \\ x & -k \le x \le k \\ -k & x < k, \end{cases} \tag{4}$$

where $k$ is a tuning constant that determines the degree of robustness; in this study, we set $k = 1.5$ [36]. The application

Table 5: Feature vector used in forecasting.

| Attribute number | Description | Formula |
|---|---|---|
| 1–24 | Hour indicator (dummy variable) | $G_i, i = 1, \ldots, 24$ |
| 25–55 | Day of the month indicator (dummy variable) | $D_i, i = 1, \ldots, 31$ |
| 56–62 | Day of the week indicator (dummy variable) | $T_i, i = 1, \ldots, 7$ |
| 63–74 | Month indicator (dummy variable) | $M_i, i = 1, \ldots, 12$ |
| 75 | Holiday indicator (dummy variable) | $S$ |
| 76 | Sunset indicator (dummy variable) | $N$ |
| 77–100 | Load of the previous 24 hours | $Z_{g-i}, i = 1, \ldots, 24$ |
| 101–104 | Minimum load of the previous 3, 6, 12, and 24 hours | $\min\{Z_{g-1}, \ldots, Z_{g-i}\}, i = 3, 6, 12, 24$ |
| 105–108 | Maximum load of the previous 3, 6, 12, and 24 hours | $\max\{Z_{g-1}, \ldots, Z_{g-i}\}, i = 3, 6, 12, 24$ |
| 109–114 | Load in the same hour of the previous week (6 days) | $Z_{g,d-i}, i = 2, \ldots, 7$ |
| 115–118 | Load in the same hour of the same day in previous weeks (4 weeks) | $Z_{g,d-i}, i = 14, 21, 28, 35$ |
| 119–122 | Average temperature observed over previous hourly periods | $\text{avg}\{T_{g-i}, \ldots, T_{g-i[+1]}\}, i = 1, 3, 6, 12, 24$ |
| 123–128 | Average temperature observed in the same hour over the previous 6 days | $T_{g,d-i}, i = 2, \ldots, 7$ |
| 129–132 | Average weekly temperature observed in the previous $i$-day periods | $\text{avg}\{T_{g,d-i}, \ldots, T_{g,d-i[+1]}\}, i = 7, 14, 21, 28, 35$ |
| 133–136 | Average humidity observed over previous hourly periods | $\text{avg}\{W_{g-i}, \ldots, W_{g-i[+1]}\}, i = 1, 3, 6, 12, 24$ |
| 137–142 | Average humidity observed in the same hour over the previous 6 days | $W_{g,d-i}, i = 2, \ldots, 7$ |
| 143–146 | Average humidity observed in the previous $i$-day periods | $\text{avg}\{W_{g,d-i}, \ldots, W_{g,d-i[+1]}\}, i = 7, 14, 21, 28, 35$ |

Notation [+1] stands for the next element from the set of indices $i$ $\{1, 3, 6, 12, 24\}$, for example, $\text{avg}\{T_{g,d-1}, \ldots, T_{g,d-3}\}$.

of this function is known as metric Winsorizing and brings extreme observations into the range $\mu \mp k$. In practice, $\sigma$ is not known; thus, a MAD robust estimator is used:

$$\text{MAD} = \text{median}\left(\left|x_i - \text{median}\left(x_i\right)\right|\right). \tag{5}$$

Finally, an accuracy measure was used to identify how many "correct" forecasts the model makes. Correctness is defined by the user, for example, forecasts within a percentage range of the actual load. However, for low loads, a percentage range may become insignificant [1]. For a load of 0.1 kWh, a 15% range would be 0.085–0.115, and a forecast of 0.2 kWh will be considered wrong, whereas such a forecast would be acceptable in practice. To address this false loss of accuracy, we set a 15% range of error to define the general accuracy, but if the load is smaller than 1 kWh, then a range of ±0.15 kWh is considered acceptable [36]. Therefore, the accuracy for hour $i$ is given by

$$\begin{aligned}\text{Accuracy} = &\sum 1\left\{L_i > 1,\ \left|L_i - P_i\right| < P_i * 0.15\right\} \\ &+ \sum 1\left\{L_i < 1,\ \left|L_i - P_i\right| < 0.15\right\}.\end{aligned} \tag{6}$$

*5.3. Predictors.* In this study, we focus on forecasting the 24-hour-ahead electricity usage. To forecast the load, we constructed a feature vector with the attributes presented in Table 5. These attributes were constructed based on time series of the hourly electricity demand. Additional variables such as temperature, humidity, and date were also collected.

Electricity demand varies depending on the time of day (daily cycles), day of the week (weekly cycles), day of the month (monthly cycles), season (seasonal cycles), and occurrence of holidays. Therefore, we enriched the analysis by considering 76 dummy variables describing the hour (1–24), day of the month (31 variables), day of the week (seven variables), month (12 variables), occurrence of holidays (one variable), and sunset in a particular hour (one variable) [1].

The main variables in the forecasting process are those derived directly from the time series, and they include the maximum, minimum, and actual demand at certain intervals. The features were created by decomposing the time series.

*5.4. Implementation.* Building predictive models involves huge volumes of data and complex algorithms. Therefore, an efficient computing environment with high-performance computers is necessary. In our case, all the numerical calculations were performed on computing clusters located at the Interdisciplinary Center for Mathematical and Computational Modelling at the University of Warsaw. The HYDRA engine with the Scientific Linux 6 operating system was used with the following nodes and parameters [1]:

(i) Istanbul: AMD Opteron™ 2435 2.6 GHz, 2 CPU × 6 cores, 32 GB RAM.

(ii) Westmere: Intel® Xeon® CPU X5660 2.8 GHz, 2 CPU × 6 cores, 24 GB RAM.

R-CRAN was used as the computing environment. This is an advanced statistical package and an interpreted programming language; it is licensed under the GNU GPL and based on the S language [36].

The starting point for the numerical experiments was the division of the *WikiEnergy* dataset into training (330 days, instead of 365 days, due to up to 35-day time window for attributes construction), validation (28 days), and testing (14 days) samples as described earlier.

The main criterion for training the models is the efficient generalization of knowledge with the least error. The most

commonly used measure to assess the quality of forecasts in the electric power system is MAPE. Therefore, to find the best parameters for all models and ensure their generalization, the following function was minimized:

$$f\left(\mathrm{MAPE}_U, \mathrm{MAPE}_V\right) = \frac{1}{2}\left|\mathrm{MAPE}_U - \mathrm{MAPE}_V\right| + \frac{1}{2}\mathrm{MAPE}_V, \tag{7}$$

where $\mathrm{MAPE}_U$ and $\mathrm{MAPE}_V$ denote the training and validation errors, respectively [36].

The experiments tested a broad set of machine learning algorithms, including artificial neural networks, regression trees, random forest regression, $k$-nearest neighbors regression, and support vector regression. In the following subsections, these algorithms are briefly introduced along with their settings. The proposed machine learning algorithms were challenged against some typical approaches used for forecasting (benchmarks), namely, naive forecast, random forecast, the ARIMA model, and stepwise regression [1].

*5.4.1. Artificial Neural Networks.* Artificial neural networks are mathematical objects in the form of equations or systems of equations, usually nonlinear, for analysis and data processing. The purpose of neural networks is to convert input data into output data with a specific characteristic or to modify such systems of equations to read useful information from their structure and parameters. On the statistical basis, selected types of neural networks can be interpreted in general nonlinear regression categories.

In studies related to forecasting in power engineering, multilayer, one-way artificial neural networks with no feedback are most commonly used. Multilayer perceptron (MLP) networks are one of the most popular types of supervised neural networks. For example, the MLP network (3, 4, 1) means a neural network with three inputs, four neurons in the hidden layer, and one neuron in the output layer. In general, the three-layer MLP neural network $(P, M, K)$ is described by the expression

$$f\left(\mathbf{x}_i, \mathbf{w}\right) = h_2\left(\mathbf{W}_2\left[h_1\left(\mathbf{W}_1\mathbf{x}_i + \mathbf{b}_1\right)\right] + \mathbf{b}_2\right), \tag{8}$$

where $\mathbf{x}_i = (x_1, \ldots, x_p)^T$ represents the input data, $\mathbf{W}_1$ is the matrix of the first layer weights with dimensions $M \times P$, $\mathbf{W}_2$ is the matrix of the second layer weights with dimensions $K \times M$, and $h_i(\mathbf{u})$ and $\mathbf{b}_i$ are nonlinearities (functions of neuron activation, e.g., logistic function) and constant values in subsequent layers, respectively.

The goal of supervised learning of the neural network is to search for such network parameters that minimize the error between the desired values $L_i$ and those received at the output of the network $P_i$. The most frequently minimized error function is the sum of the squares of differences between the actual value of the explained variable and its theoretical value determined by the model, with the values of the synaptic weight vector set:

$$E\left(\mathbf{w}\right) = \frac{1}{2}\sum_{k=1}^{K}\mathbf{e}^{(k)} = \frac{1}{2}\sum_{k=1}^{K}\left(\sum_{i=1}^{n}\left(P_i^{(k)} - L_i^{(k)}\right)^2\right), \tag{9}$$

where $n$ is the number of training samples, $P_i^{(k)}$ and $L_i^{(k)}$ are the predicted and reference values, and $K$ is the number of training epochs of the neural network.

The neural network learning process involves the iterative modification of the values of the synaptic weight vector $\mathbf{w}$ (all weights are set in one vector), in iteration $k + 1$:

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \eta_k\mathbf{p}_k, \tag{10}$$

where $\mathbf{p}_k$ is the direction of the minimization of the function $E(\mathbf{w})$ and $\eta$ is the magnitude of the learning error. The most popular optimization methods are undoubtedly gradient methods, which are based on the knowledge of the function gradient:

$$\mathbf{p}_k = -\left[\mathbf{H}\left(\mathbf{w}_k\right)\right]^{-1}\mathbf{g}\left(\mathbf{w}_k\right), \tag{11}$$

where $\mathbf{g}$ and $\mathbf{H}$ denote the gradient and the Hessian of the last known solution $\mathbf{w}_k$, respectively.

In the practical implementations of the algorithm, the exact determination of Hessian $\mathbf{H}(\mathbf{w}_k)$ is abandoned, and its approximation $\mathbf{G}(\mathbf{w}_k)$ is used instead. One of the most popular methods of learning neural networks is the algorithm of variable metrics. In this method, the Hessian (or its reversal) in each step is modified from the previous step by some correction. If by $\mathbf{c}_k$ and $\mathbf{r}_k$ the increments of the vector $\mathbf{w}$ and the gradient $\mathbf{g}$ in two successive iterative steps are marked, $\mathbf{c}_k = \mathbf{w}_k - \mathbf{w}_{k-1}$, $\mathbf{r}_k = \mathbf{g}_k - \mathbf{g}_{k-1}$, and by $\mathbf{V}_k$ the inverse matrix of the approximate Hessian $\mathbf{V}_k = [\mathbf{G}(\mathbf{w}_k)]^{-1}$, $\mathbf{V}_{k-1} = [\mathbf{G}(\mathbf{w}_{k-1})]^{-1}$, according to the most effective formula of Broyden–Fletcher–Goldfarb–Shanno (BFGS) [37], the process of updating the value of the $\mathbf{V}_k$ matrix is described by the recursive relationship

$$\mathbf{V}_k = \mathbf{V}_{k-1} + \left(1 + \frac{\mathbf{r}_k^T\mathbf{V}_{k-1}\mathbf{r}_k}{\mathbf{c}_k^T\mathbf{r}_k}\right)\frac{\mathbf{c}_k\mathbf{c}_k^T}{\mathbf{c}_k^T\mathbf{r}_k} - \frac{\mathbf{c}_k\mathbf{r}_k^T\mathbf{V}_{k-1} + \mathbf{V}_{k-1}\mathbf{r}_k\mathbf{c}_k^T}{\mathbf{c}_k^T\mathbf{r}_k}. \tag{12}$$

As a starting value, $\mathbf{V}_0 = 1$ is usually assumed, and the first iteration is carried out in accordance with the algorithm of the largest slope.

Artificial neural networks are often used to estimate or approximate functions that can depend on a large number of inputs. In contrast to the other machine learning algorithms considered in these experiments, the ANN required the input data to be specially prepared. The vector of continuous variables was standardized, whereas the binary variables were converted such that 0s were transformed into values of –1. Finally, the dependent variable was normalized by zero unitarization. To generate a forecast, the reverse transformation of the dependent variable was applied [1].

To train the neural networks, we used the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm, which is a quasi-Newton optimization method (available in the *nnet* library). The network had an input layer with 146 neurons and a hidden layer with 10, 15, 20, . . . , 50 neurons (i.e., nine sets of experiments).

A logistic function was used to activate all of the neurons in the network, and a regularization factor was introduced to penalize weights that were too high in the network (to control overfitting). Factor values of 0.01, 0.1, and 0.5 were considered [1].

In each experiment, 27 neural networks were learned with various parameters (the number of neurons in the hidden layer multiplied by the number of penalties). To avoid overfitting, after each learning iteration had finished (with a maximum of 50 iterations), the models were checked using the error measure defined in (7). Finally, out of the 27 learned networks, that with the smallest error was chosen as the best for delivering forecasts [36].

*5.4.2. k-Nearest Neighbors Regression.* The $k$-nearest neighbors regression [38] is a nonparametric method, which means that no assumptions are made regarding the model that generates the data. Its main advantage is the simplicity of the design and low computational complexity. The forecast of the value of the explained variable $L_i$ on the basis of the vector of explanatory variables $\mathbf{x}_i$ is determined as

$$P_i = \frac{\sum_{k=1}^{K} L_k I\left(\mathbf{x}_i, \mathbf{x}_k\right)}{K}, \tag{13}$$

where

$$I\left(\mathbf{x}_i, \mathbf{x}_k\right) = \begin{cases} 1, & \text{if } \mathbf{x}_k \text{ is one of the } k \text{ nearest neighbors } \mathbf{x}_i \\ 0, & \text{otherwise,} \end{cases} \tag{14}$$

whereas $\mathbf{x}_k$ is one of the $k$-nearest neighbors $\mathbf{x}_i$, in the case where the distance $d(\mathbf{x}_i, \mathbf{x}_k)$ belongs to $k$ smallest distance between the observations from the set $\mathbf{X}$ and $\mathbf{x}_k$. The most commonly used distance is the Euclid distance.

To improve the algorithm, we normalized the explanatory variables (standardization for quantitative variables and replacement of 0 by −1 for binary variables). The normalization ensures that all dimensions for which the Euclidean distance is calculated have the same importance. Otherwise, a single dimension could dominate the other dimensions [1].

The algorithm was trained with *knnreg* implemented in the *caret* library. Different values of $k$ were investigated in the experiments: {5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100, 110, 120, 130, 140, 150, 160, 170, 180, 190, 200, 250, 300}. The optimal value, and thus the final form of the model, was determined as that giving the minimum error according to (7).

*5.4.3. Regression Trees.* Regression trees are popular machine learning induction tools. They approximate target functions in a discreet way and represent them in a tree structure or alternatively in a set of decision rules. Characteristic for decision trees is the division of a multidimensional space of features into disjoint $R_k$ segments, within which a simple model approximating locally the target function within the segment is under consideration [39].

The regression tree in the *CART* version was used in this work. This tree consists of intermediate nodes in which tests are performed on explanatory variables, end nodes (leaves) storing values of the dependent variable $L_i$ (expressed as the average value of objects belonging to a given segment), and branches connecting nodes. The discussed type of model is a binary tree, which means that each branch has two branches (descendants). In the tree construction process, the intermediate sets divide the set of examples into two subsets: positive examples that meet the test assigned to the node and negative examples that do not meet this test. The aim is to ensure that the values of the target function of examples in these subsets have the smallest possible variance:

$$s^2\left(\mathbf{P}\right) = \frac{1}{N_k} \sum_{k \in R_k} \left(L_k - \overline{L}_{R_k}\right)^2, \tag{15}$$

where $N_k$ is the number of observations in the $R_k$ segment and $\overline{L}_{R_k}$ is the average value of the target function of the examples in the set $R_k$. The variance (15) is a measure of the inhomogeneity of the node (impurity). The division of the set of examples in a node into two subsets as a result of the test is justified if it leads to the reduction of heterogeneity; that is,

$$N_P s^2\left(R_{kP}\right) + N_N s^2\left(R_{kN}\right) < s^2\left(R_k\right), \tag{16}$$

where $(R_k), (R_{kP}), (R_{kN})$ are sets of indexes of examples in the parent node (parent) and in child nodes with positive or negative examples. In addition $N_P$ and $N_N$ are numbers of positive and negative examples in set $R_k$: $N_P = |R_{kP}|/|R_k|$, $N_N = |R_{kN}|/|R_k|$.

Regression trees are usually built according to the descending scheme. Starting from the start node (root) in which the entire learning set is considered, it moves to the next nodes as a result of dividing the examples into subsets $R_{kP}$ and $R_{kN}$. The size of the tree depends on the complexity of the problem under consideration. The size of the tree, which implies the degree of its fit to the learning data, can be controlled, for example, using the parameter cp. This parameter indicates to the algorithm how much the general model performance should be increased in each step, so that a given decision node can be divided.

To train regression trees, the *rpart* package was used to implement the CART algorithm. In the process of dividing a multidimensional space, the dispersion (variance) around the mean value of the dependent variable for observations belonging to the same node (leaf) was minimized. At each stage of node splitting, the variable value that minimized the sum of squares was chosen. The minimum number of observations in the node was set to 20, and the leaf was set to at least six observations; otherwise, the node was split [1].

Instead of pruning the tree at the end of the algorithm, we used pruning during the growth stage. Generally, this approach prevents new splits from being created when the previous splits provided only a slight increase in predictive accuracy. The complexity parameter (cp) varied from 0 to 0.1 in increments of 0.001, meaning that if any split did not increase the model's overall coefficient of determination by at least cp, then the split was decreed to be not worth

pursuing. The tree was built up to a depth of 30 levels. Out of 1000 regression trees that were tested, the final structure was chosen based on the error measure defined in (7) [1].

*5.4.4. Random Regression Forests.* Random forests are an ensemble learning method for regression that operate by constructing a multitude of decision trees at training time and outputting the prediction of the individual trees. Random forests correct for decision trees' habit of overfitting to their training set [40].

The training algorithm for random forests applies the general technique of bootstrap aggregating, or bagging, to tree learners. Given a training set $\mathbf{X}$ with responses $\mathbf{L}$, bagging repeatedly ($B$ times) selects a random sample with replacement of the training set and fits trees $T_i$ to these samples. After training, predictions for unseen samples $\mathbf{X}'$ can be made by averaging the predictions from all the individual regression trees on $\mathbf{X}'$:

$$\mathbf{P} = \frac{1}{B}\sum_{i=1}^{B} T_i\left(\mathbf{X}'\right). \tag{17}$$

This bootstrapping procedure leads to better model performance because it decreases the variance of the model, without increasing the bias. This means that while the predictions of a single tree are highly sensitive to noise in its training set, the average of many trees is not, as long as the trees are not correlated.

The above procedure describes the original bagging algorithm for trees. Random forests differ in only one way from this general scheme: they use a modified tree learning algorithm that selects, at each candidate split in the learning process, a random subset of the features. This process is sometimes called feature bagging. The reason for doing this is the correlation of the trees in an ordinary bootstrap sample: if one or a few features are very strong predictors for the response variable (target output), these features will be selected in many of the $B$ trees, causing them to become correlated. Typically, for a regression problem with $p$ features, $p/3$ (rounded down) features are used in each split.

To train the random regression forest, an algorithm from the *randomForest* library was used. Prior to the training, $n$-element samples were selected with replacement, and these accounted for approximately 63% of the population. The samples were used to construct the *CART* tree. Each tree was built to its maximum size (without pruning), preventing the occurrence of five or fewer observations in a leaf [1].

A randomized subset of variables was used to construct each tree. The total number of trees in the forest was 500. The final forecast was defined based on Huber's robust estimator [35]. Finally, as in previous cases, the best forest structure was chosen based on the error measure defined in (7).

*5.4.5. Support Vector Regression.* Support vector learning is based on simple ideas which originated in statistical learning theory [41]. The simplicity comes from the fact that support vector machines (SVMs) apply a simple linear method to the data but in a high-dimensional feature space nonlinearly related to the input space. Moreover, even though we can think of SVMs as a linear algorithm in a high-dimensional space, in practice, it does not involve any computations in that high-dimensional space.

SVMs use an implicit mapping $\Phi$ of the input data into a high-dimensional feature space defined by a kernel function, that is, a function returning the inner product $\langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_\mathbf{i}')\rangle$ between the images of two data points $\mathbf{x}_i$, $\mathbf{x}_\mathbf{i}'$ in the feature space. The learning then takes place in the feature space, and the data points only appear inside dot products with other points [42]. More precisely, if a projection $\Phi : \mathbf{X} \rightarrow \mathbf{H}$ is used, the dot product $\langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_\mathbf{i}')\rangle$ can be represented by a kernel function $k$:

$$k\left(\mathbf{x}_i, \mathbf{x}_\mathbf{i}'\right) = \left\langle \Phi\left(\mathbf{x}_i\right), \Phi\left(\mathbf{x}_\mathbf{i}'\right)\right\rangle, \tag{18}$$

which is computationally simpler than explicitly projecting $\mathbf{x}_i$ and $\mathbf{x}_\mathbf{i}'$ into the feature space $\mathbf{H}$.

Training an SVM regression involves solving a quadratic optimization problem. Using a standard quadratic problem solver for training an SVM would involve solving a big QP problem even for a moderate sized dataset, including the computation of an $n \times n$ matrix in memory ($n$ training points). In general, predictions correspond to the decision function

$$P_i = \operatorname{sign}\left(\left\langle \mathbf{w}, \Phi\left(\mathbf{x}_i\right)\right\rangle\right), \tag{19}$$

where solution $\mathbf{w}$ has an expansion $\mathbf{w} = \alpha_i \sum_\mathbf{i} \Phi(\mathbf{x}_i)$ in terms of a subset of training patterns that lie on the margin.

Using a different loss function called the $\varepsilon$-insensitive loss function $\|L_i - f(\mathbf{x}_i)\|_\varepsilon = \max\{0, \|L_i - f(\mathbf{x}_i)\| - \varepsilon\}$, SVMs can also perform regression. This loss function ignores errors that are smaller than a certain threshold $\varepsilon > 0$, thus creating a tube around the true output. The primal optimization problem takes the form

$$\begin{aligned}
\text{minimize} \quad & (t, \mathbf{w}) \\
& = \frac{1}{2}\|\mathbf{w}\|^2 \\
& \quad + \frac{C}{n}\sum_{i=1}^{n}\left(\xi_i + \xi_i^*\right), \\
\text{subject to} \quad & \left(\left\langle \Phi\left(\mathbf{x}_i\right), \mathbf{w}\right\rangle + b\right) - L_i \\
& \leq \varepsilon - \xi_i, \\
& L_i - \left(\left\langle \Phi\left(\mathbf{x}_i\right), \mathbf{w}\right\rangle + b\right) \\
& \leq \varepsilon - \xi_i^*, \\
& \quad \xi_i^* \geq 0, \ (i = 1, \ldots, n),
\end{aligned} \tag{20}$$

where $C$ is the cost parameter that controls the penalty paid by the SVM for misclassifying a training point and thus the complexity of the prediction function. A high cost value $C$ will force the SVM to create a complex enough prediction function to misclassify as few training points as possible, while a lower cost parameter will lead to a simpler prediction function.

To construct the support vector regression model, $\varepsilon$-SVR from the *kernlab* library with sequential minimal optimization (SMO) was used to solve the quadratic programming

problem. A linear kernel function was used, and $\varepsilon$ (which defines the margin width for which the error function is zero) was arbitrarily taken from the following set: {0.01, 0.05, 0.1, 0.25, 0.5, 0.75, 1} [1].

The regularized parameter $C$ that controls overfitting was arbitrarily set to one of the following values {0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.25, 0.5, 0.75, 1}. Finally, as in all previous cases, the model that minimized the error function (7) was chosen [1].

*5.4.6. Naive Forecast.* The naive forecast was constructed in the following manner. For the forecasting horizon of 24 h, the value recorded on the previous day at that hour was taken as the forecast [1].

*5.4.7. Random Forecast.* The random forecast was constructed in the following way. Given the electricity consumption in a given hour on a particular day of the week, empirical distribution functions were computed. Using a *runif* function, a value between 0 and 1 was then drawn from a uniform distribution ($p$ probability). This value was used to estimate the quantile of the empirical distribution (the final value of the forecast) according to a weighted averaging of the order statistic $L_i$ (quantile function):

$$Q_p = (1 - \gamma) L_i + \gamma L_{i+1}, \tag{21}$$

where $\gamma = np + m - g$, $n$ is the number of observations, $g = \text{floor}(np + m)$, and $m = 1 - p$ [1].

*5.4.8. ARIMA Model.* Autoregressive integrated moving-average model provides a description of a stationary stochastic process in terms of two polynomials, one for the autoregression and the other for the moving average that is applied in some cases where data show evidence of nonstationarity, where an initial differencing step (corresponding to the integrated part of the model) can be applied one or more times to eliminate the nonstationarity [43]. For a given time series of data $L_i$, an ARIMA$(p, d, q)$ model is given by

$$\left(1 - \sum_{j=1}^{p} \varphi_j B^j\right)(1 - B)^d L_i = \delta + \left(1 + \sum_{j=1}^{q} \theta_j B^j\right)\varepsilon_j, \tag{22}$$

where $p$ is the order (number of time lags) of the autoregressive model, $d$ is the degree of differencing (the number of times the data have had past values subtracted), $q$ is the order of the moving-average model, $B^j$ is the lag operator, $\varphi_j$ are the parameters of the autoregressive part of the model, $\theta_j$ are the parameters of the moving-average part, $\varepsilon_j$ are error terms, and finally $\delta$ is a drift factor $\delta = \mu(1 - \varphi_1 - \cdots - \varphi_p)$, where $\mu$ is the mean of $(1 - B)^d L_i$. The error terms $\varepsilon_j$ are generally assumed to be independent, identically distributed variables sampled from a normal distribution with zero mean.

To determine the order of an ARIMA model, a useful criterion is the Akaike information criterion (AIC) written as

$$\text{AIC} = -2 \log(L) + 2(p + q + k + 1), \tag{23}$$

where $L$ is the likelihood of the data; the parameter $k$ in this criterion is defined as the number of parameters in the model being fitted to the data. Therefore, this part uses the *auto.arima* function implemented in the *forecast* library. The function identifies and estimates the model by minimizing Akaike's information criterion.

In this model, the maximum values for the AR and MA orders were arbitrarily set to $p = 14$ and $q = 14$. The degree of differencing was tested with the Kwiatkowski–Phillips–Schmidt–Shin (KPSS) test, which examines the null hypothesis that an observable time series is stationary around a deterministic trend [1].

*5.4.9. Stepwise Regression.* Stepwise linear regression of the form

$$L_i = \mathbf{x}_i^T \boldsymbol{\beta} + \varepsilon_i \tag{24}$$

was used as an automated tool to identify a useful subset of predictors (where $\mathbf{x}_i$ denotes vector of independent variables, $\boldsymbol{\beta}$ is the vector of the models' parameters, and $\varepsilon_i$ is the error variable). Two procedures were tested: the first adds the most significant variable (forward selection) and the second removes the least significant variable during each step (backward elimination). The forward approach stops when all variables not in the model have $p$ values that are greater than the specified alpha-to-enter value (5% significance level was used). The backward elimination stops when all variables in the model have $p$ values that are less than or equal to the specified alpha-to-remove value (5% significance level was used) [1].

## 6. Forecasting Aggregated Electricity Demand

*6.1. Forecasting Electricity Demand for the Entire Population.* In the first step of the forecasting stage, the hourly ($g$) values of electricity demand were aggregated by summing all $n$ values of $z_g^{(i)}$ in the entire population:

$$z_g^{\text{population}} = \sum_{i=1}^{n} z_g^{(i)}. \tag{25}$$

For the clarity of the presented results, the following notation is introduced: $Z_{24}$, naive forecast; $F_g$, random forecast; ARIMA, ARIMA model; $L\_f$, stepwise regression with forward selection; $L\_b$, stepwise regression with backward elimination; $k$-NN, $k$-nearest neighbors regression; RPART, regression trees; RF, random regression forests; NNET, artificial neural network; and SVR, support vector regression [1].

Forecasting results for hourly models with a forecast horizon of 24 h are presented in Table 6. The best results for the test set were obtained by SVR, NNET, and the two comparative methods ($L\_f$ and $L\_b$). After examining each of the separate subsets (training, validation, and test) and the error metrics, it is clear that the machine learning models (except for random forests) exhibit good stability. There is also no difference between the MAPE and r-MAPE metrics, which indicates that the forecasts are largely inaccurate.

Table 6: Forecasting results for aggregate models of the entire population.

| Model | MAPE (%) | r_MAPE (%) | Acc (%) | MSE | MAPE (%) | r_MAPE (%) | Acc (%) | MSE | MAPE (%) | r_MAPE (%) | Acc (%) | MSE |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Learning sample | | | | Validation sample | | | | Test sample | | | |
| $Z_{24}$ | 16.05 | 14.77 | 57.60 | 158 | 16.06 | 15.40 | 54.38 | 75 | 20.17 | 18.82 | 46.43 | 152 |
| $F_g$ | 41.17 | 36.68 | 25.98 | 1059 | 52.30 | 44.96 | 29.87 | 917 | 49.19 | 44.30 | 25.60 | 814 |
| ARIMA | 9.48 | 9.01 | 79.71 | 45 | 19.33 | 18.89 | 45.77 | 103 | 26.28 | 25.08 | 39.00 | 186 |
| $L\_f$ | 14.76 | 13.74 | 61.31 | 118 | 18.85 | 18.80 | 41.60 | 91 | 17.38 | 16.45 | 56.85 | 102 |
| $L\_b$ | 14.70 | 13.68 | 61.74 | 118 | 19.96 | 19.91 | 36.85 | 101 | 17.85 | 16.91 | 56.55 | 103 |
| KNN | 14.64 | 13.39 | 62.12 | 131 | 14.63 | 14.20 | 60.77 | 56 | 20.41 | 19.41 | 47.02 | 127 |
| RPART | 15.58 | 14.39 | 59.60 | 133 | 16.55 | 15.48 | 56.61 | 72 | 20.46 | 19.30 | 49.40 | 153 |
| RF | 0.50 | 0.42 | 100.00 | 0 | 14.07 | 13.69 | 63.15 | 49 | 19.46 | 18.31 | 49.70 | 117 |
| NNET | 13.65 | 12.69 | 65.34 | 103 | 13.62 | 13.35 | 61.66 | 65 | 17.50 | 16.76 | 52.98 | 110 |
| SVR | 14.82 | 13.62 | 61.78 | 126 | 14.40 | 14.17 | 55.42 | 58 | 17.03 | 16.04 | 57.14 | 100 |

*6.2. Forecasting Electricity Demand within Clusters.* The hourly ($g$) values of electricity demand were then aggregated by summing all $n_k$ values of $z_g^{(i)}$ in the $k$th cluster:

$$z_g^{C_k} = \sum_{i=1}^{n_k} z_g^{(i)}. \tag{26}$$

Similar to the previous experiment, the aggregate explanatory variables were determined based on Table 5. For each of the 10 similarity measures and eight possible partitions of household aggregates (total 80 combinations), forecasting models based on machine learning algorithms were developed, as described in Section 5. A total of 440 different "definitive" forecasting models have been developed. Taking into account the number of indirect models developed during the optimization process, the number of models increases to 328680 (32(RPNN) + 102(RPART) + 12(RF) + 27(NNET) + 77(SVR) = 249).

In the next step, the forecasted demand for the whole population was determined as the sum of all $q$ predicted values $p_g^{(k)}$ for the $k$th cluster:

$$p_g^{\text{population}} = \sum_{k=1}^{q} p_g^{(k)}. \tag{27}$$

For example, let the actual aggregate value of the hourly electricity demand for the entire population at time $g$ be $z_g^{\text{population}} = 46\,\text{kWh}$. Taking into account the second split of households into clusters in terms of the autocorrelation-based distances, the predicted aggregates for these clusters are $p_g^{(1)} = 25\,\text{kWh}$, $p_g^{(2)} = 12\,\text{kWh}$, and $p_g^{(3)} = 10\,\text{kWh}$. By aggregating the above values in accordance with (11), the final forecast for the whole population is $p_g^{\text{population}} = 47\,\text{kWh}$.

To identify situations in which partitioning households into clusters improved the final forecast, we introduced a percentage point sensitivity range and denoted different cases using the following fonts [1]:

(i) Italic font: forecast improves in terms of Acc and MAPE/r-MAPE when determining the final forecast according to (11); for example, for a model with 20% error, the improvement should be at least 0.5 percentage points, so the model error should be less than 19.5%.

(ii) Bold font: forecast worsens in terms of Acc and MAPE/r-MAPE when determining the final forecast according to (11); for example, for a model with 20% error, an increase of at least 0.5 percentage points should be expected, so the model error is greater than 20.5%.

(iii) Roman font: neutral cases in which Acc and MAPE/r-MAPE remain at similar levels; for example, for a model with 20% error, we define a 1 percentage point range (19.5–20.5%) over which no improvement is observed when determining the final forecast according to (11).

Forecasts for the hourly demand models developed in this study, with a 24-hour forecast horizon, are presented in Table 7. This table compares the results of the forecasts for a particular partition against the results of the base model developed for the aggregate of all households (see Table 6).

In the majority of cases, application of $k$-NN and RPART leads to a deterioration with respect to the baseline models developed for the aggregate population and the same forecasting methods (for the second partition of the TQuest distance and the RPART model, there is some improvement). The RF approach gives improved predictions using EDR, STS, and TQuest. However, the results with RF and SVR are often neutral. The NNET forecasting method generally improves the results for each time series similarity measure (worsening results in only 5% of cases), as presented in Figure 5. Using the distance based on cross-correlation, EDR, $L_p$, LCSS, and TQuest, the forecast accuracy increased by nearly 1.5 percentage points. In general, the following pattern is observed: the MAPE error decreases when the number of clusters increases (please refer to Average and BASE_NNET labels in Figure 5). This observation suggests that the underlying population was very heterogeneous and splitting it into a number of clusters leads to improved forecast accuracy.

To provide a quantitative summary of the experiments, we tested whether there are statistically significant differences

Table 7: Forecasting results in terms of MAPE error for grouped households using the test set.

| Partition number | Autocorrelation-based distance | | | | | Partition number | Fourier coefficients-based distance | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | KNN | RPART | RF | NNET | SVR | | KNN | RPART | RF | NNET | SVR |
| 1 | **21.45** | 20.04 | 19.28 | 17.20 | 17.19 | 1 | **21.10** | 20.08 | 19.36 | **18.01** | 16.88 |
| 2 | **22.52** | 20.83 | 19.77 | *16.67** | 17.22 | 2 | **22.23** | **21.70** | 19.50 | 17.32 | 17.02 |
| 3 | **23.39** | **21.05** | 19.73 | 17.24 | 17.10 | 3 | **22.46** | **21.93** | 19.16 | 17.01 | 16.86 |
| 4 | **24.51** | **21.85** | 19.77 | 17.14 | 16.88 | 4 | **22.48** | **21.88** | 19.19 | *16.88* | 16.79 |
| 5 | **26.00** | **22.66** | 19.95 | *16.90* | 16.70 | 5 | **23.54** | **22.78** | 19.28 | 17.41 | *16.39* |
| 6 | **26.50** | **22.32** | 19.44 | *16.49** | 16.64 | 6 | **23.99** | **21.98** | 19.28 | *16.69** | 16.56 |
| 7 | **27.26** | **22.09** | 19.28 | *16.05** | *16.51* | 7 | **25.53** | **22.23** | 19.18 | *16.94* | 16.71 |
| 8 | **28.68** | **21.96** | 19.13 | *16.25** | 16.55 | 8 | **25.72** | **22.49** | 19.00 | 17.33 | 16.63 |
| Partition number | Distance based on cross-correlation | | | | | Partition number | Periodogram-based distance | | | | |
| | KNN | RPART | RF | NNET | SVR | | KNN | RPART | RF | NNET | SVR |
| 1 | **21.19** | 20.24 | 19.74 | 17.20 | 16.88 | 1 | **20.94** | 20.87 | 19.47 | 17.61 | 17.42 |
| 2 | **21.45** | 20.43 | 19.52 | 17.58 | 17.02 | 2 | **22.34** | **21.23** | 19.59 | *16.75* | 16.81 |
| 3 | **21.78** | 20.64 | 19.24 | 17.32 | 17.06 | 3 | **23.30** | **21.62** | 19.37 | 17.11 | 16.81 |
| 4 | **21.64** | 20.77 | 19.19 | *16.99* | 16.90 | 4 | **24.08** | 20.61 | 19.15 | 17.09 | 16.74 |
| 5 | **21.73** | 20.56 | 19.26 | *15.65** | 16.84 | 5 | **25.50** | **21.22** | 19.48 | *16.59** | *16.30** |
| 6 | **22.21** | **21.00** | **18.90** | *16.32** | 16.92 | 6 | **27.19** | **21.90** | 19.57 | *16.86* | *16.16** |
| 7 | **22.51** | 20.30 | 19.21 | *16.99* | 16.90 | 7 | **27.53** | **21.69** | 19.46 | *16.67** | *16.07** |
| 8 | **22.48** | 20.42 | 19.15 | *15.50** | 16.87 | 8 | **27.73** | **21.81** | 19.60 | 17.31 | *16.09** |
| Partition number | DTW distance | | | | | Partition number | LCSS distance | | | | |
| | KNN | RPART | RF | NNET | SVR | | KNN | RPART | RF | NNET | SVR |
| 1 | **21.10** | 20.08 | 19.36 | **18.01** | 16.88 | 1 | **20.92** | **21.09** | 19.18 | **19.35** | **17.57** |
| 2 | **22.23** | **21.70** | 19.50 | 17.32 | 17.02 | 2 | **21.19** | **21.17** | 19.08 | 17.24 | 17.45 |
| 3 | **23.34** | **21.58** | 19.44 | *16.52** | 17.06 | 3 | **22.79** | **22.36** | *18.91* | *16.81* | 17.07 |
| 4 | **23.57** | **21.85** | 19.44 | *16.22** | 16.90 | 4 | **23.44** | **21.04** | 19.06 | 17.31 | 16.95 |
| 5 | **23.59** | **21.82** | 19.44 | 17.36 | 16.84 | 5 | **24.26** | **20.99** | 19.18 | *15.95** | 16.72 |
| 6 | **24.88** | **22.66** | 19.30 | *16.84* | 16.92 | 6 | **25.09** | **21.07** | 19.08 | *16.58** | 16.66 |
| 7 | **24.95** | **22.51** | 19.28 | 17.27 | 16.90 | 7 | **26.38** | 20.70 | 19.15 | *16.40** | *16.47* |
| 8 | **26.68** | **22.78** | 19.14 | *16.47** | 16.87 | 8 | **27.06** | 20.61 | 19.32 | *16.32** | *16.41* |
| Partition number | EDR | | | | | Partition number | STS distance | | | | |
| | KNN | RPART | RF | NNET | SVR | | KNN | RPART | RF | NNET | SVR |
| 1 | 20.85 | **21.00** | 19.06 | *16.95* | 17.30 | 1 | **20.92** | **21.52** | 19.44 | 17.24 | 17.49 |
| 2 | **22.15** | **21.02** | *18.73** | 17.31 | 17.29 | 2 | **21.32** | **21.62** | 19.22 | *16.90* | 17.17 |
| 3 | **23.39** | **21.08** | 18.96 | *16.57** | 16.94 | 3 | **21.80** | **21.06** | *18.81* | *16.92* | 17.00 |
| 4 | **24.26** | **21.31** | 18.99 | *15.98** | 16.78 | 4 | **22.15** | **22.77** | 18.99 | 17.17 | 17.05 |
| 5 | **24.67** | **21.51** | 18.98 | *16.53** | 16.71 | 5 | **22.10** | **22.30** | *18.88* | 17.01 | 17.18 |
| 6 | **25.95** | **21.99** | *18.69** | *16.57** | 16.77 | 6 | **22.41** | **21.78** | *18.92* | *16.76** | 16.83 |
| 7 | **25.92** | **21.77** | *18.50** | *16.54** | 16.76 | 7 | **22.44** | **21.80** | *18.79* | *16.56** | 16.79 |
| 8 | **27.86** | **22.02** | 19.01 | *16.83* | 16.68 | 8 | **23.12** | **21.41** | *18.85* | *16.26** | 16.73 |
| Partition number | $L_p$ distance | | | | | Partition number | TQuest distance | | | | |
| | KNN | RPART | RF | NNET | SVR | | KNN | RPART | RF | NNET | SVR |
| 1 | **21.10** | 20.08 | 19.36 | **18.01** | 16.88 | 1 | **21.08** | 20.15 | 19.27 | 17.29 | 16.99 |
| 2 | **22.23** | **21.70** | 19.50 | 17.32 | 17.02 | 2 | **21.48** | *19.78* | *18.88* | *16.36** | 16.54 |
| 3 | **22.46** | **21.93** | 19.16 | 17.01 | 16.86 | 3 | **21.70** | 20.01 | 19.09 | *16.58** | *16.41* |
| 4 | **22.48** | **21.88** | 19.19 | *16.88* | 16.79 | 4 | **21.88** | 20.04 | *18.86* | *16.48** | *16.32** |
| 5 | **23.74** | **22.43** | 19.26 | 17.03 | *16.34* | 5 | **22.55** | 20.84 | *18.85* | 17.44 | *16.35* |
| 6 | **24.35** | **22.03** | 19.23 | *16.02** | *16.50* | 6 | **23.02** | **21.60** | *18.57** | *15.77** | 16.60 |
| 7 | **24.59** | **22.61** | 19.32 | *16.38** | 16.56 | 7 | **23.09** | **21.65** | *18.64** | *16.03** | 16.55 |
| 8 | **26.33** | **22.41** | 19.05 | *15.85** | 16.61 | 8 | **23.19** | **21.69** | *18.41** | *16.05** | *16.47* |

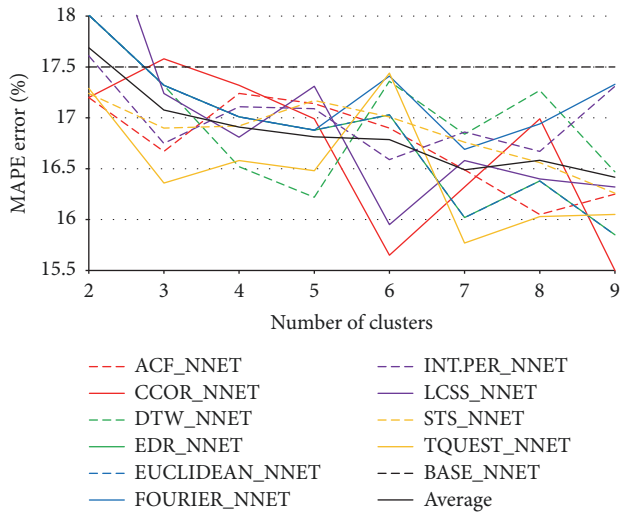


Figure 5: MAPE errors for NNET with respect to the number of clusters and the similarity measure.

in terms of the MAPE errors when forecasting the usage of the entire population against forecasting in segments. The Kolmogorov-Smirnov (K-S) test was used to compare two empirical error distributions with respect to the datasets. The differences were found to be significant at $p < 0.05$ and marked as $*$ in Table 7.

Considering all the partitions that improve the results for NNET (the choice of this forecasting method results from generating the smallest errors), some measures of the optimal number of clusters give the partitions leading to the optimal improvement in prediction quality. The accuracy of these measures is as follows:

(i) Calinski and Harabasz index: 20%.

(ii) Duda index: 40%.

(iii) $C$ index: 50%.

(iv) Ptbiserial index: 40%.

(v) DB index: 80%.

(vi) Frey index: 0%.

(vii) Hartigan index: 20%.

(viii) Ratkowsky index: 20%.

(ix) Ball index: 20%.

(x) McClain index: 10%.

(xi) KL index: 10%.

(xii) Silhouette index: 50%.

(xiii) Dunn index: 40%.

(xiv) SD index: 70%.

The above percentages were calculated according to the number of well-defined subpartitions giving a quality improvement for each similarity measure. For example, the Duda index using the distance based on the cross-correlation, EDR, periodogram-based distances, and LCSS distance gives the optimal number of clusters as 8, 7, 8, and 6, respectively (please refer to Table 8). By applying these values to Table 7 (underlined italic font), it can be seen that they overlap with an improvement in forecasts (four correct cases divided by 10 time series similarity measures).

## 7. Conclusions

The main objective of this article was to develop an effective method for dealing with short-term forecasts of electricity demand for the whole population and for clusters of households over 24-hour forecasting horizons based on hourly data. To achieve this goal, we investigated the following:

(i) Partitioning of households into disjoint clusters using a hierarchical algorithm based on ten time series similarity measures.

(ii) Determination of the optimum number of clusters for each time series similarity measure.

(iii) Application of different techniques for short-term forecasting using aggregated data and within the groups of homogenous households (segments).

Experiments were designed to answer the research questions concerning load forecasting for a group of customers. In particular, it can be concluded that:

(i) the number of households included in each cluster is very diverse, taking into account the same type of partition and among different time series similarity measures;

(ii) the distances based on cross-correlation, STS, and TQuest give the most unbalanced cluster sizes;

(iii) the results of time series clustering can be divided according to the degree of similarity; we can distinguish a group of four measures with a high degree of similarity (DTW, $L_p$, Fourier coefficients, and periodogram-based distances) and very low (autocorrelation-based distances and LCSS distance) similarity;

(iv) the optimal number of groups based on the majority vote indicated that the households should be divided into two groups;

(v) there are groups of measures for optimal clustering, indicating a similar number of groups for particular groups of time series similarities;

(vi) models developed based on machine learning algorithms show good stability and can be used in practice to forecast aggregates for the entire population;

(vii) of all the developed methods, neural network models are characterized by the best results for aggregates of the entire population for hourly data;

(viii) the use of artificial neural networks for group forecasting offers, in most cases, increased accuracy of forecasts in relation to the aggregate population;

(ix) for neural networks, the following general pattern is observed: the forecasting error decreases when the number of clusters increases;

Table 8: Results for the relevant number of clusters based on the partition type and the time series similarity measure.

| | KL | CH | Hartigan | Cindex | DB | Silhouette | Duda | Ratkowsky | Ball | Ptbiserial | Frey | McClain | Dunn | SDindex |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| acf | 5 | 4 | 4 | 2 | 9 | 2 | 2 | 4 | 3 | 3 | 2 | 2 | 9 | 2 |
| ccor | 2 | 2 | 9 | 9 | 6 | 5 | 8 | 2 | 3 | 6 | 1 | 4 | 9 | 2 |
| dtw | 3 | 2 | 3 | 4 | 2 | 2 | 3 | 2 | 3 | 2 | 3 | 2 | 2 | 6 |
| edr | 3 | 2 | 3 | 2 | 2 | 2 | 7 | 3 | 3 | 3 | 1 | 2 | 3 | 2 |
| euclidean | 2 | 2 | 3 | 4 | 5 | 2 | 4 | 2 | 3 | 2 | 2 | 2 | 2 | 5 |
| fourier | 2 | 2 | 3 | 3 | 5 | 2 | 4 | 2 | 3 | 2 | 2 | 2 | 2 | 5 |
| int.per | 9 | 2 | 3 | 8 | 2 | 2 | 8 | 2 | 3 | 2 | 2 | 2 | 9 | 3 |
| lcss | 5 | 5 | 5 | 2 | 9 | 9 | 6 | 8 | 3 | 9 | 1 | 2 | 9 | 9 |
| sts | 2 | 2 | 5 | 2 | 9 | 9 | 2 | 6 | 3 | 9 | 1 | 2 | 9 | 8 |
| tquest | 7 | 7 | 6 | 9 | 5 | 4 | 2 | 7 | 3 | 6 | 1 | 2 | 6 | 3 |

(x) there are groups of measures that select the optimal number of clusters more often than others, with partitions leading to improved forecast quality.

Finally, note that this research was carried out on a sample of 46 households. The conclusions serve as an indication of how the specific measures of time series similarity and optimal clustering, as well as forecasting methods, could behave on a much larger scale.

When load shape clustering is used as a preprocessing step for subsequent household-level segmentation, the requirement for a low number of clusters, which is practical for tariff purposes, should be relaxed to capture the diverse time-of-use behavior in residential hourly consumption. In addition, regarding the size distribution and visual examination conducted in the postcluster checking, future work should evaluate the broader validation of the robustness of the identified clusters using additional data sources, such as household demographic and socioeconomic information and rate structures. In particular, the sources of variability in the identified hourly patterns need to be established and better understood to support more effective demand-side management and behavior-based programs [44].

## Conflicts of Interest

The authors declare that they have no conflicts of interest.

## Acknowledgments

This study was cofounded by the National Science Centre, Poland, Grant no. 2016/21/N/ST8/02435.

## References

[1] K. Gajowniczek and T. Zabkowski, "Electricity forecasting on the individual household level enhanced based on activity patterns," *PLoS ONE*, vol. 12, no. 4, Article ID e0174098, 2017.

[2] G. Chicco, "Overview and performance assessment of the clustering methods for electrical load pattern grouping," *Energy*, vol. 42, no. 1, pp. 68–80, 2012.

[3] L. Jin, D. Lee, A. Sim et al., *Comparison of Clustering Techniques for Residential Energy Behavior using Smart Meter Data*, Lawrence Berkeley National Lab. (LBNL), Berkeley, CA, USA, 2017.

[4] P. Montero and J. A. Vilar, "TSclust: An R package for time series clustering," *Journal of Statistical Software* , vol. 62, no. 1, pp. 1–43, 2014.

[5] U. Mori, A. Mendiburu, and J. A. Lozano, "Distance measures for time series in R," *The TSdist Package R journal*, vol. 8, no. 2, pp. 455–463, 2016.

[6] J. Mei, M. Liu, Y.-F. Wang, and H. Gao, "Learning a Mahalanobis Distance-Based Dynamic Time Warping Measure for Multivariate Time Series Classification," *IEEE Transactions on Cybernetics*, vol. 46, no. 6, pp. 1363–1374, 2016.

[7] E. Bianco-Martinez, N. Rubido, C. G. Antonopoulos, and M. S. Baptista, "Successful network inference from time-series data using mutual information rate," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 26, no. 4, Article ID 043102, 9 pages, 2016.

[8] P. Berkhin, "A survey of clustering data mining techniques," in *Grouping Multidimensional Data*, J. Kogan, C. Nicholas, and M. Teboulle, Eds., pp. 25–71, Springer, Berlin, Germany, 2006.

[9] D. Xu and Y. Tian, "A comprehensive survey of clustering algorithms," *Annals of Data Science*, vol. 2, no. 2, pp. 165–193, 2015.

[10] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, "On clustering validation techniques," *Journal of Intelligent Information Systems*, vol. 17, no. 2-3, pp. 107–145, 2001.

[11] Pecanstreet, "Dataport," 2014, https://dataport.pecanstreet.org/.

[12] B. Yi and C. Faloutsos, "Fast time sequence indexing for arbitrary *Lp* norms," in *Proceedings of the 26th International Conference on Very Large Data Bases (VLDB '00)*, pp. 385–394, San Francisco, Calif, USA, 2000.

[13] C. S. Möller-Levet, F. Klawonn, K. Cho, and O. Wolkenhauer, "Fuzzy clustering of short time-series and unevenly distributed sampling points," in *Advances in Intelligent Data Analysis V*, vol. 2810 of *Lecture Notes in Computer Science*, pp. 330–340, Springer, Berlin, Germany, 2003.

[14] D. Lin, W. Gu, Y. Wang, X. Yuan, Q. Li, and R. Wang, "Synthetic evaluation of power quality based on dynamic time warping spatial distance measurement," *Power System Technology*, vol. 2, p. 047, 2013.

[15] L. Chen, M. T. Özsu, and V. Oria, "Robust and fast similarity search for moving object trajectories," in *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD '05)*, pp. 491–502, ACM, June 2005.

[16] M. Vlachos, G. Kollios, and D. Gunopulos, "Discovering similar multidimensional trajectories," in *Proceedings of the 18th International Conference on Data Engineering*, pp. 673–684, San Jose, Calif, USA, March 2002.

[17] P. D'Urso and E. A. Maharaj, "Autocorrelation-based fuzzy clustering of time series," *Fuzzy Sets and Systems*, vol. 160, no. 24, pp. 3565–3589, 2009.

[18] J. Caiado, N. Crato, and D. Peña, "A periodogram-based metric for time series classification," *Computational Statistics & Data Analysis*, vol. 50, no. 10, pp. 2668–2684, 2006.

[19] R. Tolimieri, M. An, and C. Lu, *Algorithms for Discrete Fourier Transform and Convolution*, Springer Science & Business Media, 2013.

[20] J. Aßfalg, H.-P. Kriegel, P. Kröger, P. Kunath, A. Pryakhin, and R. Renz, "Similarity search on time series based on threshold queries," in *Advances in Database Technology—EDBT*, vol. 3896 of *Lecture Notes in Computer Science*, pp. 276–294, Springer, Berlin, Germany, 2006.

[21] M. Charrad, N. Ghazzali, V. Boiteau, and A. Niknafs, "Nbclust: An R package for determining the relevant number of clusters in a data set," *Journal of Statistical Software*, vol. 61, no. 6, pp. 1–36, 2014.

[22] T. Zabkowski and K. Gajowniczek, "Grade analysis for households segmentation based on energy usage patterns," in *Proceedings of the 2017 IEEE International Conference on INnovations in Intelligent SysTems and Applications (INISTA '17)*, pp. 168–173, July 2017.

[23] S. Saraçli, N. Doğan, and I. Doğan, "Comparison of hierarchical cluster analysis methods by cophenetic correlation," *Journal of Inequalities and Applications*, vol. 1, p. 203, 2013.

[24] T. Zabkowski, K. Gajowniczek, and R. Szupiluk, "Grade analysis for energy usage patterns segmentation based on smart meter data," in *Proceedings of the 2nd IEEE International Conference on Cybernetics (CYBCONF '15)*, pp. 234–239, June 2015.

[25] F. Javed, N. Arshad, F. Wallin, I. Vassileva, and E. Dahlquist, "Forecasting for demand response in smart grids: an analysis on use of anthropologic and structural data and short term multiple loads forecasting," *Applied Energy*, vol. 96, pp. 150–160, 2012.

[26] U. Dent, U. Aickelin, and T. Rodden, "The application of a data mining framework to energy usage profiling in domestic residences using UK data," in *Proceedings of the Research Students Conference on Buildings Dont Use Energy, People Do? Domestic Energy Use and CO2 Emissions in Existing Dwellings*, pp. 1–10, Bath, UK, 2011.

[27] Y. G. Yohanis, J. D. Mondol, A. Wright, and B. Norton, "Real-life energy use in the UK: how occupancy and dwelling characteristics affect domestic electricity use," *Energy and Buildings*, vol. 40, no. 6, pp. 1053–1059, 2008.

[28] H. K. Alfares and M. Nazeeruddin, "Electric load forecasting: literature survey and classification of methods," *International Journal of Systems Science*, vol. 33, no. 1, pp. 23–34, 2002.

[29] G. E. P. Box, G. M. Jenkins, and G. C. Reinsel, *Time Series Analysis: Forecasting and Control*, Prentice Hall, Englewood Cliffs, NJ, USA, 3rd edition, 1994.

[30] Z. Aung, J. Williams, A. Sanchez, M. Toukhy, and S. Herrero, "Towards Accurate Electricity Load Forecasting in Smart-Grids," in *Proceedings of the The 4th International Conference on Advances in Databases, Knowledge and Data Applications, DBKDA, SaintGilles*, pp. 51–57, Reunion Island, France, 2012.

[31] K. Amasyali and N. El-Gohary, "Building lighting energy consumption prediction for supporting energy data analytics," *Procedia Engineering*, vol. 145, pp. 511–517, 2016.

[32] G. Mitchell, S. Bahadoorsingh, N. Ramsamooj, and C. Sharma, "A comparison of artificial neural networks and support vector machines for short-term load forecasting using various load types," in *Proceedings of the 2017 IEEE Manchester PowerTech (Powertech '17)*, pp. 1–4, IEEE, Manchester, UK, June 2017.

[33] W. Kong, Z. Y. Dong, D. J. Hill, F. Luo, and Y. Xu, "Short-term residential load forecasting based on resident behaviour learning," *IEEE Transactions on Power Systems*, vol. 33, no. 1, pp. 1087-1088, 2018.

[34] K. Gajowniczek and T. Zabkowski, "Short term electricity forecasting based on user behavior from individual smart meter data," *Journal of Intelligent & Fuzzy Systems: Applications in Engineering and Technology*, vol. 30, no. 1, pp. 223–234, 2016.

[35] M. Huber, M. Lechner, and C. Wunsch, "The performance of estimators based on the propensity score," *Journal of Econometrics*, vol. 175, no. 1, pp. 1–21, 2013.

[36] K. Gajowniczek and T. Zabkowski, "Two-stage electricity demand modeling using machine learning algorithms," *Energies*, vol. 10, no. 10, article 1547, 2017.

[37] N. Andrei, "An adaptive scaled BFGS method for unconstrained optimization," *Numerical Algorithms*, vol. 77, no. 2, pp. 413–432, 2018.

[38] B. Nguyen, C. Morell, and B. De Baets, "Large-scale distance metric learning for k-nearest neighbors regression," *Neurocomputing*, vol. 214, pp. 805–814, 2016.

[39] R. A. Berk, "Classification and regression trees (CART)," in *Statistical Learning from A Regression Perspective*, pp. 129–186, Springer, 2016.

[40] J. Huo, T. Shi, and J. Chang, "Comparison of Random Forest and SVM for electrical short-Term load forecast with different data sources," in *Proceedings of the 7th IEEE International Conference on Software Engineering and Service Science (ICSESS '16)*, pp. 1077–1080, IEEE, August 2016.

[41] F. Davò, M. T. Vespucci, A. Gelmini, P. Grisi, and D. Ronzio, "Forecasting Italian electricity market prices using a Neural Network and a Support Vector Regression," in *Proceedings of the 2016 AEIT International Annual Conference (AEIT '16)*, pp. 1–16, IEEE, October 2016.

[42] K. Muandet, K. Fukumizu, B. Sriperumbudur, and B. Schölkopf, "Kernel mean embedding of distributions: A review and beyond," *Foundations and Trends in Machine Learning*, vol. 10, no. 1-2, pp. 1–141, 2017.

[43] R. J. Hyndman and Y. Khandakar, "Automatic time series forecasting: the forecast package for R," *Journal of Statistical Software* , vol. 27, no. 3, pp. 1–22, 2008.

[44] J. Ling, D. Lee, A. Sim et al., *Comparison of Clustering Techniques for Residential Energy Behavior using Smart Meter Data*, AAAI Workshops & Artificial Intelligence for Smart Grids and Buildings, San Francisco, Calif, USA, 2017.