Nearest Neighbors-Based Forecasting for Electricity Demand Time Series in Streaming

L. Melgar-García[1(✉)], D. Gutiérrez-Avilés[2], C. Rubio-Escudero[2], and A. Troncoso[1]

[1] Data Science and Big Data Lab, Pablo de Olavide University, 41013 Seville, Spain
{lmelgar,atrolor}@upo.es
[2] Department of Computer Science, University of Seville, Seville, Spain
{dgutierrez3,crubioescudero}@us.es

Abstract. This paper presents a new forecasting algorithm for time series in streaming named StreamWNN. The methodology has two well-differentiated stages: the algorithm searches for the nearest neighbors to generate an initial prediction model in the batch phase. Then, an online phase is carried out when the time series arrives in streaming. In par-ticular, the nearest neighbor of the streaming data from the training set is computed and the nearest neighbors, previously computed in the batch phase, of this nearest neighbor are used to obtain the predictions. Results using the electricity consumption time series are reported, show-ing a remarkable performance of the proposed algorithm in terms of fore-casting errors when compared to a nearest neighbors-based benchmark algorithm. The running times for the predictions are also remarkable.

Keywords: Forecasting · Nearest neighbors · Streaming time series · Electricity demand

1 Introduction

The explosive increase of global data, based on technology improvements, has led to the gathering of information as an automatic and relatively inexpensive task [16], taking us to the big data era. Data science offers a solution to gain knowledge from these enormous amounts of data, by means of adapting the existing models to the big data paradigm. This adaptation is a challenge for the research community.

There are several fields in which the application of the new big data anal-ysis techniques represent a great improvement in problem solving, such as the energy consumption forecasting [17,25]. Governments and private companies are focusing on this topic as the improvement in the prediction levels will have both economic and environmental positive consequences [22]. In this sense, some classifiers have already been successfully applied to electricity consumption forecast

[24], such as the weighted k-nearest neighbors classifier (WKNN). The WKNN [4] is a generalization of the k-nearest neighbors method (KNN) [2] that assigns weights to the neighbors based on their distance from the element to predict.

The direct application of these methods to the big data domain is not feasible due to the computational needs, in terms of time and memory. Several proposals have adapted nearest neighbor proposals to the big data paradigm using the Apache Spark distributed computation framework [16,21,22].

Data streams are generated in many practical applications as temporally ordered, fast changing and massive flows of data [13]. Mining these data streams is concerned with extracting knowledge structures represented in models and patterns in non-stopping streams of information [6], and the research on this area has gained a high attraction. In this proposal, we go a step further and propose a general purpose forecasting algorithm based on nearest neighbors for big volumes of streams of data, to create a method capable to be integrated in real-world systems in which data are constantly generated as streams, such as the demand prediction in the electricity market.

In this work, we propose the StreamWNN algorithm for streaming time series forecasting based on nearest neighbors. This algorithm consists of two phases: a batch phase to generate an initial model, and an online phase for forecasting in real time by using the model previously created in the batch phase. The proposal has been applied to a dataset of 497,832 samples of electrical energy consumption in Spain.

The rest of the paper is structured as follows. Section 2 describes a review of the state of the art approaches related to data streaming and forecasting analysis int the electricity market. Section 3 presents the methodology applied for time series forecasting in streaming. The experimental setup along with the results obtained using electricity demand time series can be found in Sect. 4. Finally, in Sect. 5, the final considerations extracted from this work are presented.

2 Related Works

A wide range of approaches for data streaming analysis is currently emerging. The primary trend of this research field is the development of machine learning methodologies to the streaming environments. From this perspective, the authors in [26] presented an online version of the support vector machine model to predict air pollutant levels from the monitored air pollutant in Hong Kong. An online version of the linear discriminant analysis algorithm for dimension reduction was presented in [14]. On the other hand, it has carried out research efforts to develop frameworks for adaptation of standard machine learning methods to streaming [10]. Another streaming framework is SAMOA presented in [1], where the authors developed an API to apply machine learning algorithms to streams of data in a big data context. Different algorithms to analyze data streams from the Internet of Things (IoT) networks are also currently being developed. In [5], the authors presented a streaming linear regression method to forecast streams data generated by IoT networks. Finally, several surveys have been published

about the streaming analysis. In this sense, the authors in [20] analyzed the difference between the real-time processing and the stream processing of big data, and by contrast, a survey of the open-source technologies that support big data in a real-time/near real-time environments was introduced in [15].

Concerning researches focused on forecasting for the electricity demand time series data, in addition to the nearest neighbors for a big data environment proposed in [22] and the new big data-based multivariate and multi-output forecasting approach in [21], other approaches have been published. In [7], the authors applied decision gradient boosted trees and random forest ensemble methods to the electricity demand problem. Also, deep learning techniques have been applied to predict energy power consumption in big data environments [23]. A Temporal Convolutional Network has been used in [11] for demand energy forecasting. A complete review of deep learning architectures for time series forecasting was published in [12]. On the other hand, several streaming techniques have been applied to this problem. In [3], the authors presented an incremental pattern characterization algorithm to mine data streams from smart meters of RMIT University for the purpose of applying it to electricity consumption analysis and forecasting. The authors in [8] proposed a complete data streaming analysis system combining an online clustering model and neural-networks to predict in real-time the electricity load demand from sensor networks.

Besides the forecasting, other problems related to energy have been addressed. The authors in [19] presented a methodology to extract electric energy consumption patterns in big data time series based on the application of the distributed version of the k-means algorithm. In [9] the authors presented a big data system to classify fraudulent behaviors of the leading electricity company in Spain. Regarding the streaming environment, an incremental ensemble learning method is developed for the on-line classification of the electricity pricing in Australia in [18]. Furthermore, in [27], the authors presented the DStreamEPK algorithm, a new streaming clustering method applied to electric power data.

3 Methodology

This Section presents the proposed algorithm, named StreamWNN, for streaming time series forecasting based on nearest neighbors.

The time series forecasting problem consists in predicting the next h values from the historical past values. The StreamWNN forecasting algorithm has of two phases: a batch phase to generate an initial model, and an online phase for forecasting in real time by using the model created in the batch phase.

A time series X_t is defined as a set of ordered chronologically values $\{x_1, ..., x_t\}$ and can be always transformed into N instances formed by features and class as follows:

$$X_t = \{(x^1, y^1), ..., (x^N, y^N)\} \quad x^i \in \mathbb{R}^w \quad y^i \in \mathbb{R}^h \tag{1}$$

where x^i are the features of the $i-th$ instance, representing the past w values to the class y^i formed by the next h values. For the batch phase, the time series X_t

from Eq. (1) is divided into training set and test set. Then, the prediction method based on nearest neighbors searches for the k closest neighbors to a window composed of the past w values to the h values to be predicted. Afterwards, a weight is calculated for each neighbor depending on its distance to the past values window. Thus, the initial model M consists of the pairs of the features of the instances from the test set and a list of the classes corresponding to the neighbors of theses features from the training set. That is:

$$M = <x^i, <y(n_1(x^i)), ..., y(n_K(x^i))>> \tag{2}$$

where K is the number of neighbors, x^i are the w features of the $i-th$ instance of the test set, $n_j(x^i)$ is the $j-th$ neighbor of the x^i and $y(n_j(x^i))$ is the class corresponding to the $j-th$ neighbor.

When a time series is received in streaming, a temporal data stream ds_t can be a chunk of the time series of length w, that is, $ds_t = <x_t, x_{t+1}, ..., x_{t+w-1}>$. For the online phase, once the ds_t data stream is received, the nearest neighbor of the ds_t from test set is obtaining by this equation:

$$x^* = arg \min_{x^i \in Test} d(x^i, ds_t) \tag{3}$$

Then, the prediction is obtained using the K neighbors of x^* and weights already computed in the M model from Eq. (2). In particular, the prediction is made by applying a weighted average of the h samples following those k closest neighbors. Thus, the StreamWNN algorithm predicts by means of the following equation:

$$\widehat{y}(ds_t) = \frac{1}{\sum_{j=1}^{K} w_j^*} \sum_{j=1}^{K} w_j^* y(n_j(x^*)) \tag{4}$$

where $n_j(x^*)$ is the $j-th$ neighbor of x^*, $y(n_j(x^*))$ is the class corresponding to the $j-th$ neighbor, and w_j^* is the weight associated to the $j-th$ neighbor. This weight depends on the distance, with a greater weight to the closest neighbors and a smaller weight to the farthest neighbors according to a distance d. In this work, the Euclidean distance has been chosen, and the weights are defined by:

$$w_j^* = \frac{1}{d^2(x^*, n_j(x^*))} \tag{5}$$

Consequently, it is possible to obtain forecasts in real time as the prediction consists of making an average with neighbors and weights previously computed in the batch phase using the historical data.

4 Experimental Results

This section specifies the dataset used in the experimentation and reports the results obtained after the application of the proposed streaming algorithm. In particular, Sect. 4.1 describes the dataset and the experiments carried out, specifying in each case the parameters of the algorithm. Finally, in Sect. 4.2 the results of the experimentation are shown and discussed.

4.1 Dataset and Experimental Setup

The experimentation uses a dataset of 497,832 samples of electrical energy consumption in Spain. Each sample has 12 attributes related to electricity. For this work, only two attributes are used: the energy demand in megawatt (MW) and the date and time of the measured value.

In particular, the dataset contains 1 sample for every 10 min during 9 years and 6 months, starting the 1 January 1^{st} 2007 and finishing June 21^{st} 2016. The whole dataset is chronologically divided into 3 sets of data: training, test and streaming sets. The training and test sets are approximately a 70% of the dataset: the training set contains data from January 1^{st} 2007 to August 23^{rd} 2011 and the test set contains data from August 24^{th} 2011 to August 19^{th} 2013. The algorithm predicts almost 3 years, i.e., the streaming set is from August 20^{th} 2013 to June 21^{st} 2016.

In this study, the experiments are carried out with the same parameters and prediction horizons established in [22]. Each of the four experiments has a different horizon: 4, 8, 12 and 24 hours. As the dataset contains 1 sample each 10 min, the prediction horizons are 24, 48, 72 and 144 samples, respectively. The goal is to analyze the behaviour of the algorithm for different prediction horizons considering the optimal parameters of [22].

The parameters for each experiment are listed below, where h is the prediction horizon, w corresponds to the number of past values used for predicting the next h values and K is the number of nearest neighbors of the training set to consider when creating the M model, as defined in Sect. 3:

- For the prediction horizon $h = 24$, optimal parameters are $w = 144$ and $K = 4$.
- For the prediction horizon $h = 48$, optimal parameters are $w = 288$ and $K = 2$.
- For the prediction horizon $h = 72$, optimal parameters are $w = 576$ and $K = 4$.
- For the prediction horizon $h = 144$, optimal parameters are $w = 864$ and $K = 4$.

4.2 Results

The four experiments are run on a cluster located at the Data Science and Big Data Laboratory in Pablo de Olavide University. The cluster is formed by 4 nodes: 3 slaves and 1 master. The whole cluster has 4 Processors Intel(R) Core(TM) i7-5820K CPU with 48 cores, 120 GB of RAM memory. It uses Ubuntu 16.04.1 LTS, Apache Spark 2.3.4, HDFS on Hadoop 2.7.7 and Apache Kafka 2.11.

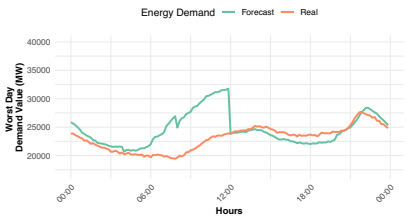
The metrics used to evaluate the performance of the algorithm are the mean absolute percentage error (MAPE), expressed as a percentage, and the mean absolute error (MAE), expressed in MW [24]. Table 1 presents the above-mentioned metrics of error when forecasting the streaming set of data for the different prediction horizons. Moreover, the maximum, minimum and standard deviation (st. dev.) of the MAPE for the streaming set are depicted. It can be noticed that both MAPE and MAE increase with higher values of the prediction horizon. Considering that in this work the offline summary model is not updated,

the standard deviation and values of MAPE and MAE lead to think that the offline summary model represents in an accurate way the streaming data.

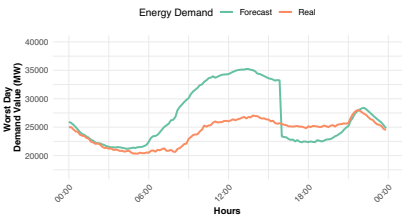
Table 1. Metrics of errors for different prediction horizons

h	w	k	Maximum MAPE	Minimum MAPE	St. dev. MAPE	MAPE	MAE
24	144	4	33.0031	0.2464	2.0745	2.4288	670.1298
48	288	2	31.2719	0.4101	2.0842	2.7617	766.8640
72	576	4	34.3861	0.6002	2.8199	3.3535	933.9924
144	864	4	29.3277	0.6548	3.6136	3.8465	1072.8357

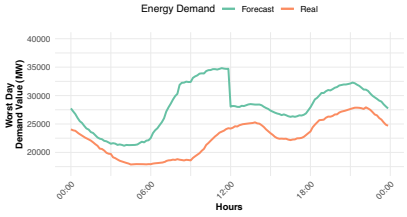
Figures 1 and 2 show the worst forecasts (the maximum MAPE) and the best ones (the minimum MAPE) for each prediction horizon, respectively. They both show the real and forecasted electricity demand values in the vertical axis and the hours of the day in the horizontal axis. Each sub-figure includes the day (in format day/month/year) and the horizon of the maximum or minimum MAPE. All worst days correspond to public holidays in Spain: in summer for the prediction horizons 24 and 48 and, in winter for the prediction horizons 72 and 144. For prediction horizons 24, 48 and 72, it can be observed abrupt changes at the last time sample of the horizon as the following forecasted values correspond to the next prediction horizon on the same day. On the other hand,



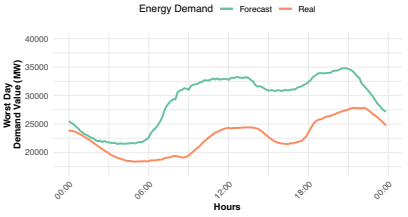
(a) h=24. Horizon: from 08:00AM to 11:50AM. Day: 16/08/2015



(b) h=48. Horizon: from 08:00AM to 15:50PM. Day: 16/08/2014



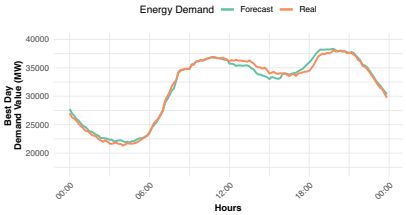
(c) h=72. Horizon: from 00:00AM to 11:50AM. Day: 25/12/2013



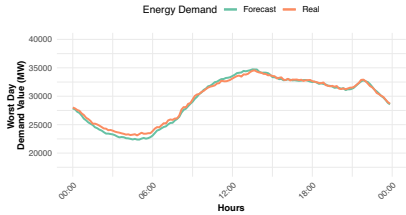
(d) h=144. Horizon: The whole day. Day: 25/12/2014

Fig. 1. Days with the worst forecasts for each h horizon

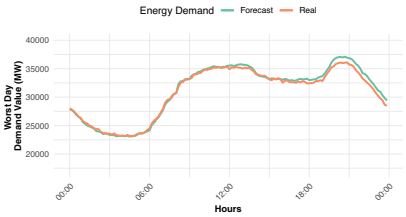
Fig. 2 shows that, in these days, the data used in the offline phase represents well the online data because even without any update of the summary offline model, the forecasted values are quite accurate.



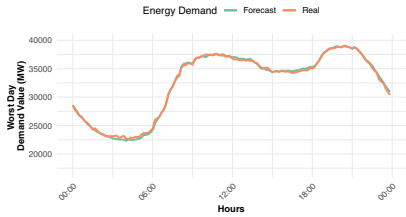
(a) h=24. Horizon: from 08:00AM to 11:50AM. Day: 20/01/2014



(b) h=48. Horizon: from 16:00PM to 23:50PM.Day: 11/08/2015



(c) h=72. Horizon: from 00:00AM to 11:50AM. Day: 18/03/2015



(d) h=144. Horizon: The whole day. Day: 26/01/2015

Fig. 2. Days with the best forecasts for each h horizon

Table 2 shows the MAE obtained when applying the algorithm recently published in [22] and the proposed StreamWNN algorithm using the same set of data and the same parameters for comparison purposes. It can be observed that the error of the proposed algorithm is higher just for $h = 24$. However, the MAEs of the StreamWNN are quite smaller than the ones in [22] for all the other prediction horizons.

Table 2. The MAE (in MW) for the StreamWNN and the algorithm in [22].

h	[22]	StreamWNN
24	524.14	670.13
48	920.87	766.86
72	1313.40	933.99
144	1514.92	1072.84

Figure 3 represents the mean values for each hour, both of the forecasted and of the real energy demand values of the $h = 24$ prediction horizon setup. The

representation of the other three forecast horizons is very similar. It confirms that the forecast results have behave very similar to the ones of the real data.

Besides the good performance, a streaming algorithm has to provide timely results during the online phase. Even if the offline phase of the streaming algorithm is not limited in execution time, the offline phase of the proposed algorithm is fast considering the huge amount of data, both in training and test sets. The offline phase of the proposed algorithm for $h = 24$ takes 222.09 s, 167.16 s for $h = 48$, 153.47 s for $h = 72$ and 122.50 s for $h = 144$.

The online execution time for all four prediction horizons is presented in Fig. 4. This figure shows for every 200 iterations of the algorithm, the time in seconds from the beginning of the online phase. The number of iterations for each experiment is different as w and h changes. In addition, as smaller these values are, less time is taken to compute the iterations (as in the offline phase). It can be observed that the algorithm increases linearly the execution time as more iterations have been previously made, which is very important in streaming algorithms. Considering these results, a forecast of h values is made in an average of 1.4 s for $h = 24$, 1.6 s for $h = 48$, 1.9 s for $h = 72$ and 2.3 s for $h = 144$. These results are presented in Table 3.

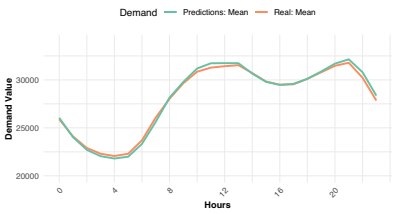


Fig. 3. Hourly average of the actual and forecasted energy demand

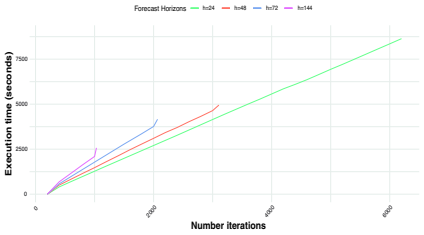


Fig. 4. Execution time of the online phase versus number of iterations

Table 3. Computation times (in seconds) for different prediction horizons

h	Offline phase time	Online prediction time of h values
24	222.09	1.4
48	167.16	1.6
72	153.47	1.9
144	122.50	2.3

5 Conclusions

The StreamWNN algorithm for time series forecasting in the streaming environment has been proposed. The StreamWNN consists of two stages: an offline or

batch phase and an online phase. The first stage creates a summary prediction model with the K nearest neighbors for each window of w samples and their next h samples of the training set. Afterwards, in the second stage, the time series of the streaming set are processed satisfying the streaming requirements. When streams arrive, the model predicts the h next values with a weighted average using the selected K nearest neighbor from the batch prediction model. The algorithm has been applied to an electricity demand time series dataset containing records over nine years. The performance of the algorithm has been evaluated with the MAPE and MAE error metrics for each prediction horizon. A good performance has been shown when comparing these errors with a benchmark algorithm, that used the same dataset and parameters.

The future works will be focused on some characteristics of the algorithm such as updating the summary batch model considering the knowledge of the previous time series streams, detecting novelties and outliers in the streams or studying the process to select the optimal values of the parameters.

Acknowledgements. The authors would like to thank the Spanish Ministry of Science, Innovation and Universities for the support under project TIN2017-88209-C2.

References

1. Bifet, A., Morales, G.F.: Big data stream learning with SAMOA. In: Proceedings of the IEEE International Conference on Data Mining Workshop (ICDM), pp. 1199–1202 (2015)
2. Cover, T., Hart, P.: Nearest neighbor pattern classification. IEEE Trans. Inf. Theory **13**(1), 21–27 (1967)
3. De Silva, D., Yu, X., Alahakoon, D., Holmes, G.: Incremental pattern characterization learning and forecasting for electricity consumption using smart meters. In: Proceedings of the IEEE International Symposium on Industrial Electronics, pp. 807–812 (2011)
4. Dudani, S.A.: The distance-weighted k-nearest-neighbor rule. IEEE Trans. Syst. Man Cybern. **6**(4), 325–327 (1976)
5. Fernández, A.M., Gutiérrez-Avilés, D., Troncoso, A., Martínez-Álvarez, F.: Real-time big data analytics in smart cities from LoRa-based IoT networks. In: Martínez Álvarez, F., Troncoso Lora, A., Sáez Muñoz, J.A., Quintián, H., Corchado, E. (eds.) SOCO 2019. AISC, vol. 950, pp. 91–100. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-20055-8_9
6. Gaber, M.M., Zaslavsky, A., Krishnaswamy, S.: Mining data streams: a review. ACM SIGMOD Rec. **34**(2), 18–26 (2005)
7. Galicia, A., Talavera-Llames, R., Troncoso, A., Koprinska, I., Martínez-Álvarez, F.: Multi-step forecasting for big data time series based on ensemble learning. Knowl. Based Syst. **163**, 830–841 (2019)
8. Gama, J., Rodrigues, P.P.: Stream-based electricity load forecast. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) PKDD 2007. LNCS (LNAI), vol. 4702, pp. 446–453. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74976-9_45

9. Gutiérrez-Avilés, D., et al.: SmartFD: a real big data application for electrical fraud detection. In: de Cos Juez, F., et al. (eds.) HAIS 2018. LNCS, vol. 10870, pp. 120–130. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-92639-1_11
10. He, H., Chen, S., Li, K., Xu, X.: Incremental learning from stream data. IEEE Trans. Neural Networks **22**(12), 1901–1914 (2011)
11. Lara-Benítez, P., Carranza-García, M., Luna-Romera, J.M., Riquelme, J.C.: Temporal convolutional networks applied to energy-related time series forecasting. Appl. Sci. **10**(7), 2322 (2020)
12. Lara-Benítez, P., Carranza-García, M., Riquelme, J.C.: An experimental review on deep learning architectures for time series forecasting. Int. J. Neural Syst. **31**(03), 2130001 (2021)
13. Li, Y., Li, D., Wang, S., Zhai, Y.: Incremental entropy-based clustering on categorical data streams with concept drift. Knowl. Based Syst. **59**, 33–47 (2014)
14. Liu, L.P., Jiang, Y., Zhou, Z.H.: Least square incremental linear discriminant analysis. In: Proceedings of the IEEE International Conference on Data Mining, pp. 298–306 (2009)
15. Liu, X., Iftikhar, N., Xie, X.: Survey of real-time processing systems for big data. In: Proceedings of the International Database Engineering and Applications Symposium, pp. 356–361 (2014)
16. Maillo, J., Ramírez, S., Triguero, I., Herrera, F.: kNN-IS: an iterative spark-based design of the k-nearest neighbors classifier for big data. Knowl. Based Syst. **117**, 3–15 (2017)
17. Martínez-Álvarez, F., Troncoso, A., Riquelme, J.C., Aguilar-Ruiz, J.S.: Energy time series forecasting based on pattern sequence similarity. IEEE Trans. Knowl. Data Eng. **23**(8), 1230–1243 (2010)
18. Ng, W.W.Y., Zhang, J., Lai, C.S., Pedrycz, W., Lai, L.L., Wang, X.: Cost-sensitive weighting and imbalance-reversed bagging for streaming imbalanced and concept drifting in electricity pricing classification. IEEE Trans. Ind. Inform. **15**(3), 1588–1597 (2019)
19. Pérez-Chacón, R., Luna-Romera, J.M., Troncoso, A., Martínez-Álvarez, F., Riquelme, J.C.: Big data analytics for discovering electricity consumption patterns in smart cities. Energies **11**(3), 683 (2018)
20. Shahrivari, S.: Beyond batch processing: towards real-time and streaming big data. Computers **3**(4), 117–129 (2014)
21. Talavera-Llames, R., Pérez-Chacón, R., Troncoso, A., Martínez-Álvarez, F.: MV-kWNN: a novel multivariate and multi-output weighted nearest neighbours algorithm for big data time series forecasting. Neurocomputing **353**, 56–73 (2019)
22. Talavera-Llames, R., Pérez-Chacón, R., Troncoso, A., Martínez-Álvarez, F.: Big data time series forecasting based on nearest neighbours distributed computing with spark. Knowl. Based Syst. **161**, 12–25 (2018)
23. Torres, J.F., Galicia, A., Troncoso, A., Martínez-Álvarez, F.: A scalable approach based on deep learning for big data time series forecasting. Integr. Comput. Aided Eng. **25**(4), 335–348 (2018)
24. Troncoso, A., Riquelme-Santos, J.M., Gómez-Expósito, A., Martínez-Ramos, J.L., Riquelme-Santos, J.C.: Electricity market price forecasting based on weighted nearest neighbors techniques. IEEE Trans. Power Syst. **22**(3), 1294–1301 (2007)
25. Troncoso, A., Riquelme, J.C., Aguilar-Ruiz, J.S., Riquelme-Santos, J.M.: Evolutionary techniques applied to the optimal short-term scheduling of the electrical energy production. Eur. J. Oper. Res. **185**(3), 1114–1127 (2008)

26. Wang, W., Men, C., Lu, W.: Online prediction model based on support vector machine. Neurocomputing **71**(4–6), 550–558 (2008)
27. Zhang, X., Qian, Z., Shen, S., Shi, J., Wang, S.: Streaming massive electric power data analysis based on spark streaming. In: Li, G., Yang, J., Gama, J., Natwichai, J., Tong, Y. (eds.) DASFAA 2019. LNCS, vol. 11448, pp. 200–212. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-18590-9_14