Received November 18, 2019, accepted December 4, 2019, date of publication December 9, 2019, date of current version December 23, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2958383

Machine Learning Algorithms for Short-Term Load Forecast in Residential Buildings Using Smart Meters, Sensors and Big Data Solutions

SIMONA-VASILICA OPREA AND ADELA BÂRA
Department of Economic Informatics and Cybernetics, Bucharest University of Economic Studies, 010374 Bucharest, Romania

Corresponding author: Simona-Vasilica Oprea (simona.oprea@csie.ase.ro)

This work was supported by the Romanian National Authority for Scientific Research and Innovation, CCCDI—UEFISCDI, through the project Multi-Layer Aggregator Solutions to Facilitate Optimum Demand Response and Grid Flexibility, under Contract 71/2018 (COFUND-ERANET-SMARTGRIDPLUS-SMART-MLA-1, within PNCDI III).

ABSTRACT In this paper, we propose a scalable Big Data framework that collects the data from smart meters and weather sensors, pre-processes and loads it into a NoSQL database that is capable to store and further process large volumes of heterogeneous data. Then, a set of Machine Learning (ML) algorithms are designed and implemented to determine the load profiles and forecast the electricity consumption for residential buildings for the next 24 hours. For the Short-Term Load Forecast (STLF), a Feed-Forward Artificial Neural Network (FF-ANN) algorithm with backtracking adjustment of the learning rate that extends and optimizes the Nesterov learning method is proposed. Its performance is compared with six algorithms, i.e. FF-ANN with well-known learning methods, namely Momentum and Nesterov, Non-linear AutoRegressive with eXogenous (NARX), Deep Neural Network (DNN), Gradient Tree Boosting (GTB) and Random Forests (RF) that are competitive and powerful ML algorithms which have been successfully used for load forecast. Hence, for STLF, the seven algorithms are executed simultaneously and the best one is automatically selected considering its accuracy in terms of Root Mean Square Errors (RMSE). The proposed methodology contains the steps required to implement the Big Data framework, i.e. data pre-processing, transformation and loading, the configuration of the ML algorithms for dimensionality reduction, clustering, STLF with different algorithms from which the Best Performant Algorithm (BPA) is automatically selected to provide STLF for the next 24 hours. The methodology is ultimately tested considering a real case of a residential smart building.

INDEX TERMS Load forecast, machine learning, Nesterov, clustering, big data.

I. INTRODUCTION

Future electricity grids consist in heterogeneous interconnected systems with an increasing number of small-scale generators and consumption appliances, providing large amounts of data. Hence, the electricity sector necessitates Big Data solutions and architectures for a performant energy system management. A diversity of statistical and artificial intelligence methods has been developed and applied for STLF. However, as proven in literature, there is no single method that exhaustively fulfil the requirements in terms of STLF since target areas vary in size, in combination of commercial, residential and industrial consumers, in geographic, climatologic and socio-economic characteristics. Furthermore, a new challenge is arising considering the large amount of available input data that continuously flows from smart meters and other sensors.

The STLF plays a key role as the consumption of the residential consumers has gained increasing quotas from the total consumption. For grid operators, STLF is essential in terms of network configuration, voltage control, and dispatching generation units. Also, the electricity consumers become more active and interested in minimizing the electricity bill. The electricity suppliers perform more accurate STLF enhancing the market strategies and settlement. Moreover, for prosumers, consumption in correlation with volatility of distribution generation (wind turbines, photovoltaic panels, etc.) leads to new challenges.

The associate editor coordinating the review of this manuscript and approving it for publication was Mostafa Rahimi Azghadi.

Most of the times, seasonal data, such as temperature, humidity and nebulosity are variables that directly influences STLF. Also, the current progress of sensors (for motion, doors position or furnace state, weather stations, etc.) provides large volume of data that could be integrated to understand the consumers' behaviour [1] and improve STLF. The influence of the weekday might be also significant due to different activities that are scheduled for a specific day. The first generation of the load forecast methods (also called analytical methods) includes time series analysis, regression methods [2]–[4], similar day method, Wavelet Transform (WT) [5], [6], least square estimation [7], etc. Artificial intelligence methods, also known as the second generation of the load forecast methods, mainly comprises ANN [8], [9], including deep neural networks [10]–[16], random forests, gradient boosting [17], fuzzy logic [18]–[20], Support Vector Machines (SVM) [21], genetic algorithms, Particle Swarm Optimization (PSO) [22], ant colony optimization-based methods [23], etc. The second generation compared with the first one has gained importance due to errors reduction. Some combination of methods (known as hybrid methods) that belong to both generations is also possible [24]–[28].

The importance of the load forecast especially after the unbundling of the energy sectors and different approaches for STLF underlying the ANN methods are emphasized in [29]. STLF of buildings in microgrids enhancing renewables integration and economics of market transactions is proposed in [24]. Load forecast using smart metering and sensors data takes into account efficient peak management measures [30]. STLF methods cover a wide range of approaches that focus on ANN methods [31] for building load forecast [32]. As ANN algorithms have been shaded by the overfitting problem, [33] propose a performant ANN with a learning method that implements a novel search technique avoiding the overfitting. An ANN method with backpropagation algorithm considers three approaches in terms of confidence intervals for performing STLF [34]. Various ANN methods for STLF of buildings in the context of smart grids underlying their advantages and disadvantages are described in [35]. The probabilistic electricity consumption forecast approach is based on prediction intervals that are developed by ANN models [36] using PSO to set the parameters of ANN. Effects of temperature on residential loads were found to be evident, while temperature effects on commercial and industrial loads weren't quite evident. This conclusion led to the fact that in case of a mix of consumer types, low accuracy could appear when weather (i.e. temperature) dependency assumptions are taken [37]. Forecasting accuracy, as measured with Mean Average Percentage Error (MAPE), improves considering a critical load that depends on the forecasting method. Thus, for Support Vector Regression (SVR) and FF-ANN, the critical load is reached for a group of consumers of only 200, respectively 500 [38].

Different types of ANN are reviewed in [39], such as: FF-ANN, Radial Basis Function (RBF) [40], self-organizing, feedback (recurrent), fuzzy, stochastic, probabilistic ANN, etc. setting stages and strengthening their applicability in economics predictions. Reference [41] proposes a forecast architectural model with ANN for microgrids. On the other hand, [42] identify the limitation of ANN and propose a STLF method for holidays and days that precede or follow holidays consisting in a combination of ANN and fuzzy inference method providing high accuracy of the hourly consumption forecast. Reference [43] forecasts the consumption in Bangladesh using various models, including fuzzy extension of ANN that proved to be very efficient. Forecasting load for NEPOOL region from New England using hourly temperature, humidity and electricity load with ANN, using Mean Average Error (MAE), Mean Squared Error (MSE), MAPE and daily peak forecast error to assess the accuracy is given in [44]. A day-ahead load forecast in Turkish power system based on FF-ANN, considering as input: date type, hour, temperature of the four major cities and last day load, is devised in [45]. Also, a 24-hour-ahead STLF for Turkey's Power System, with ANN, WT and ANN, WT and RBF, empirical mode decomposition and RBF, considers only historical data due to the irrelevance of temperature in terms of a very wide area [46].

Analysing the electricity consumers' behaviour, [47] perform load forecast based on the online sequential extreme machine learning applied to clusters of electricity consumers that are aggregated to finally obtain the total system load, using smart metering data of households from Ireland. The FF-ANN architecture and Levenberg-Marquardt training algorithm for electricity consumption forecasting of residential consumers provide good performance. Simulations imply one year and a half historical records for 93 households, in Portugal (Lisbon) showing that the model is reliable for forecasting and load profile [48]. In addition, a study for load forecasting, using a Big Data approach to find out the optimal number of lagged hours in regression models that would maximize the forecast accuracy, presents the fact that the naive models are not useful for benchmark purposes due to low accuracy issues [49]. With simulations performed on a set of buses from different areas: urban, sub-urban and industrial, [50] develop a bus load forecast model for day-ahead and hour-ahead prediction based on ANN using a clustering technique that leads to some hybrid forecast models characterized by high level of parameterization and efficiency. Considering data from smart meters and other sensors, a mixed ANN approach for STLF using NARX is integrated into a cloud solution that can be provided as a service for grid operators and residential consumers [51].

Recent developments in terms of STLF are underlying deep learning methods described in [52] as "one of the most promising techniques in advanced data analytics" and [53], i.e. Recurrent Neural Networks (RNN), specifically Long Short-Term Memory (LSTM) [54]–[58] and Convolutional Neural Network (CNN) [59], initially extensively used for traffic forecast [57] and image processing [60] that involve multiple layers to progressively extract higher level features from the input dataset.

Reference [52] compares deep learning with conventional ANN, but with increased complexity in terms of layers and training process. One very interesting conclusion is that the optimized number of layers for a short-term building cooling load prediction showed that the learning model do not actually need a deep architecture and a two-hidden-layer architecture worked well. Also, [61] claim that ANN with two hidden layers is enough to model most of the functions in the real world.

RNN, CNN and ARIMA models are compared from different point of view (accuracy, computational efficiency, generalizability). Forecasting the consumption of different types of buildings with a data set of one year and records at 1-hour resolution, [14] show that deep learning outperforms the ARIMA model, stressing that CNN may be the solution for future challenges in terms of day-ahead load forecast.

The RNN, such as LSTM and Sequence-to-Sequence (S2S) LSTM are proposed for building load forecast in [55], proving that S2S LSTM performs well for both one-hour- and one-minute-resolution forecast, whereas LSTM performs well only for one-hour-resolution forecast. High accuracy of STLF with DNN and two-stage ensemble strategy is obtained in [62]. Deep learning, namely factored conditional restricted Boltzmann machine is proposed for estimating the building electricity consumption [63], outperforming other methods, such as SVR, RNN, etc.

The performance of any load forecasting method is heavily influenced by the volume and quality of data. For this reason, smart meters and sensors play a key role since they enable appropriate decisions based on the valuable information that comes out of data. Itis a consequence of the grid operators' decision to implement smart meters at large scale. Therefore, conventional meters that are monthly read are replaced by smart meters that can hourly or at a higher resolution transmit the consumption. Thus, large volumes of data will be generated by smart meters. Also, for STLF or load profiles, more data sources are needed to correlate the consumption with other significant factors like weather conditions, web surveys or questionnaires. In this context, heterogeneous sources need to be collected and processed and new informatics solutions are required for an efficient data management. Big Data that refers to the management of large volumes of data represents the solution for data storage, processing and analytics. Big Data is characterized by at least three "V" dimensions: Volume, Velocity, and Variety. For instance, the *volume* of data collected from smart meters at 15 minutes form a city/region with 1 million of electricity consumers is around 7 GB per day considering approx. 30 bytes per reading in case of the following attributes: timestamp, meter id, active power, reactive power. In addition, the data collected from weather sensors/web APIs and other sensors may be added to correlate the consumption with other significant factors (e.g. temperature, wind speed, cloud cover). Regarding *variety*, in case of STLF and load profiles, the data sources are heterogeneous, collected in various formats and structures: raw, JSON or CSV from smart meters and sensors; JSON, CSV or XML from web APIs, document oriented (JSON, XML) from web surveys or questionnaires. As for *velocity*, data is collected usually at 15 minutes, but for ultra-short-term forecast or monitoring and control activities, the readings may be collected even more frequently (minute by minute).

In this paper, we concentrate on a scalable Big Data solution that collects data from smart meters and weather sensors, pre-processes and loads it into a NoSQL database that is capable to store various data formats without laborious data normalization as required by the relational databases. Then, a set of ML algorithms are designed and implemented to provide STLF for the next 24 hours based on the data stored in the NoSQL database. These algorithms are linked into a framework that provides an automatic process flow for determining the load profiles and day-ahead forecast. The proposed methodology contains the step by step procedure to implement this framework in the context of Big Data, i.e. data pre-processing, transformation and loading, the configuration of the ML algorithms for dimensionality reduction, clustering, load forecast with different ML algorithms to automatically choose the best performant method. Thus, the methodology is transparent, flexible, integrated and could be easily applied in any environment or data sets. The original elements of the proposed methodology consist in:

- Dynamically selecting the most significant weather attributes that influence the electricity consumption by applying three feature selection algorithms. Therefore, the proposed approach can be applied in various regions, automatically detecting the relevant meteorological factors that will be considered as input, reducing the time and resources for additional investigations related to attributes identification. Also, by selecting significant predictors, the ANN dimensionality is reduced, the accuracy is increased due to the noise filtering of uncorrelated attributes. The process required for data analysis is performed automatically, no additional step being required;
- Clustering the electricity consumption for each season considering the similar consumption behaviour. Apart from consumption data, useful insights from web surveys or questionnaires may be extracted and processed. By grouping the consumers in relevant clusters and obtaining the load profiles, their consumption pattern becomes more predictable compared to the total aggregated consumption. Also, load profiles provide a more accurate perspective over the structure of the electricity consumption by underlining their contribution to peak and off-peak. Therefore, by estimating the electricity consumption for each cluster, the forecast errors will be reduced and the STLF performance will be improved;
- Designing and developing a FF-ANN algorithm based on Nesterov learning method in which backtracking is introduced to adjust the learning rate, thus reducing the convergence rate and providing an optimum step for the minimization of the forecast errors. For evaluating its

accuracy, the proposed algorithm is compared with other six ML algorithms, i.e. two FF-ANN with Momentum and classic Nesterov learning methods, NARX, DNN, GTB, RF;
- Developing a scalable Big Data framework for STLF that processes data and simultaneously executes the seven ML algorithms to automatically select the best performant one to provide the output for the day-ahead load forecast. The BPA is selected by comparing the forecast accuracy of the seven algorithms for the last 24 hours in terms of RMSE, average, minimum and maximum values of the errors.

This paper is structured in four sections. In the current section, the significance of STLF, Big Data concept and different consumption forecast approaches from literature are presented. In section 2, we describe the proposed methodology considering the following three stages: data gathering (from smart meters and sensors), pre-processing and storage; dimensionality reduction and clustering; STLF with several ML algorithms. In section 3, we run several simulations and provide the results starting from data management to STLF performance evaluation. In section 4, the main conclusions are drawn.

II. METHODOLOGY

The proposed methodology for STLF in the Big Data context consists in three stages that imply data pre-processing, validation, storing, further processing and analysis with ML algorithms to determine the load profiles and STLF.

In the first stage, data is gathered from smart metering devices, combined with meteorological data that was collected from the weather sensors or web APIs. In order to validate the readings, an Extract, Transform and Load (ETL) process is applied and then the data is loaded into a NoSQL database. Regarding the type of the NoSQL database, key-value or document-oriented types are recommended considering the structure of the data sources. In our simulations, since all data sources are in JSON format, we use MongoDB to store data as Binary JSON (BSON) that is a binary-encoded serialization of the JSON format. MongoDB provides faster reading speed and is better suited for rapid growth when the structure of the data sources is not clearly known from the beginning (compared with another NoSQL database, such as CouchDB).

In the second stage, data analysis is performed to identify the most significant attributes that influence the electricity consumption. The attributes regarding weather parameters, type of the day (working/weekend) or season are automatically ranked based on their influence on the electricity consumption. Also, a clustering method using *k-means* is run to group the electricity consumers into consumption groups with similar behaviour. Apart from the consumption data, other optional sources may be included for clustering, such as web surveys or questionnaires. Both clusters and the most significant attributes are considered as input for the ML algorithms.

In the third stage, the STLF for the next 24 hours is performed. As a novelty, an FF-ANN algorithm is proposed with an enhanced learning method by introducing backtracking for adjusting the learning rate and reducing the computational time especially in case of large data sets. To compare its performance, other six powerful and competitive ML algorithms are implemented, i.e. FF-ANN with enhanced gradient descend methods (Momentum and Nesterov), DNN, ensemble algorithms (RF and GTB) and NARX. To obtain the best results, in the proposed methodology, all seven ML algorithms are simultaneously executed and the best performant algorithm in terms of accuracy is automatically selected to forecast the electricity consumption. The ML algorithms are trained and incrementally validated at regular intervals (monthly) to obtain a more accurate prediction. The algorithm that provides the best accuracy in terms of RMSE on the previous 24-hour forecast is automatically selected to perform the day-ahead forecast. The ML algorithms are implemented in Python as an integrated package for STLF.

The stages are linked and executed in the following sequence - stage 1: data gathering, pre-processing and storage in the NoSQL database; stage 2: clustering, dimensionality reduction; stage 3: short-term load forecast (Figure 1).

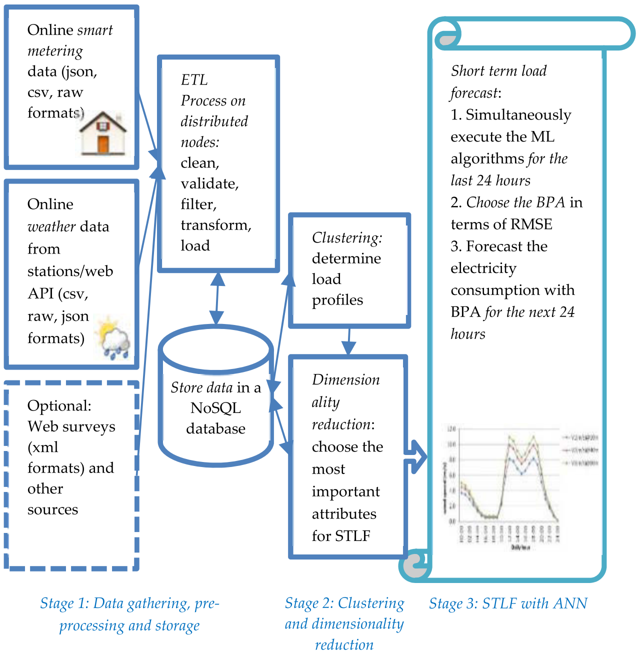
Online *smart metering* data (json, csv, raw formats)
Online *weather* data from stations/web API (csv, raw, json formats)
Optional: Web surveys (xml formats) and other sources
ETL Process on distributed nodes: clean, validate, filter, transform, load
Store data in a NoSQL database
Clustering: determine load profiles
Dimensionality reduction: choose the most important attributes for STLF
Short term load forecast:
1. Simultaneously execute the ML algorithms *for the last 24 hours*
2. *Choose the BPA* in terms of RMSE
3. Forecast the electricity consumption with BPA *for the next 24 hours*
Stage 1: Data gathering, pre-processing and storage
Stage 2: Clustering and dimensionality reduction
Stage 3: STLF with ANN

FIGURE 1. Proposed methodology for STLF in the Big Data context.

These stages are briefly described in Table 1.

A. DATA GATHERING, PRE-PROCESSING AND STORAGE

One of the biggest challenges is to process and analyse large amounts of data from online sources, generated by smart meters and sensors while data is still "alive", has value and impact. However, online data stream processing needs filtering, transformation and aggregation in order to provide time-based analyses. Data assessment can be seen

TABLE 1. Stages and steps of the proposed methodology.

Stage	Inputs	Steps
Stage 1: Data gathering, pre-processing and storage	*Data sources: Smart meters online data*; *Weather data from sensors* or *web API:* temperature, wind speed, wind bearing, humidity, dew point, visibility, cloud cover, pressure, precipitation; Optional: for load profiles, other data sources from *web surveys or questionnaires* may be included.	1. Pre-process data to clean, filter and transform it; 2. Store data into a NoSQL database (MongoDB, Elasticsearch, Oracle NoSQL).
Stage 2: Clustering and dimensionality reduction	For *clustering*: seasonal or monthly aggregated hourly consumption data and, optional, useful insights extracted from web surveys or questionnaires that may contribute to the clustering process; For *dimensionality reduction*: Aggregated consumption data at 10-15 minutes interval correlated with weather data.	1. Determine load profiles with *k-means clustering* algorithm; 2. Three *feature selection* methods rank the input factors. Considering their results, the most significant attributes that are influencing the electricity consumption are automatically identified.
Stage 3: STLF with ML algorithms	Aggregated consumption for each cluster at 10-15 minutes interval correlated with the most significant attributes (weather, season, type of the day, hour). Parameters of the ML algorithms, such as: learning rate, training method, number of hidden layers, number of neurons on the hidden layer, training epochs, value of the accepted error, activation function, number of predictors, max depth, etc. depending on the algorithm.	*Training and validation:* 1. Input standardization; 2. Split data set for training and validation; 3. Set up the parameters for the ML algorithms; 4. Initialize weights and biases; 5. Simultaneously execute the ML algorithms; 6. Automatically select the BPA based on accuracy (RMSE); 7. Incremental validation and testing. *Day-ahead forecast:* 1. Simultaneously execute the ML algorithms for the last 24 hours; 2. Automatically select the BPA based on accuracy (RMSE); 3. Run the BPA to forecast the electricity consumption for the next 24 hours.

as a filter that retains the significant data, identifies events and discards irrelevant data. Different smart meters, weather sensors or web APIs provide variate formats and recording rates that might be too detailed for certain analysis, therefore aggregation is necessary. Another challenge is related to the data quality since the data generated by smart meters or sensors could be missing, inconsistent or incomplete due to damages or communication interruptions [64]. When data quality issue is persistent, major prejudice to the STLF accuracy is caused. In order to eliminate these major risks, several steps should be performed for data pre-processing and transformation.

The smart meters installed at residential consumers usually record electricity consumption data at 1 up to 60 minutes, containing the following information: timestamp, meter id, active power (electricity consumption), reactive and apparent power. As mentioned before, weather conditions have a major influence on the electricity consumption, therefore weather data should be included as input. Accurate weather data and forecast may be obtained from weather sites as web APIs or from the weather stations located in the proximity of the consumption place (apartments, houses). The weather data usually has a 60 minute-resolution and contains values related to temperature, wind speed, wind bearing, humidity, dew point, visibility, cloud cover, pressure, precipitation intensity and probability. The measurements recorded by smart meters and weather sensors need a proper validation, filtering and transformation process to be valuable and consistent for the STLF. Thus, an ETL process is applied on these data sources and the missing or corrupted values are replaced by the most appropriate values from the previous time intervals using backward interpolation. In case of weather APIs, the non-numerical values (e.g. icon or sky-description and cloud-cover) require to be transformed into numerical values using encoders and converters. Optionally, in case of pre-processing data from other web sources (surveys, questionnaires or even social media), more advanced pre-processing methods should be applied after loading and storing data.

Since data gathered from smart meters, weather sensors / web APIs and web sources contain semi-structured data in various formats and with different time resolution, then the solution for data processing and storage could be a NoSQL database with powerful search engines, such as: MongoDB, Elasticsearch, Oracle NoSQL, Riak or Redis.

B. CLUSTERING AND DIMENSIONALITY REDUCTION

The electricity consumption data from regions with similar meteorological conditions is split over seasons and, for each consumer (c_l), the hourly average is computed. Then, the consumers with similar consumption behaviour are grouped into load profiles using the *k-means* algorithm. *K-means* is an unsupervised machine learning algorithm that builds clusters by grouping instances (consumers) around k centroids by computing the distance or the similarity between these instances. The distance between two instances is determined with one of the following classical methods: Euclidean, Manhattan or Mahalanobis. Initially, a centroid value for each cluster (μ_k) is randomly chosen, then the following steps are performed iteratively:

1) determine the distance between each instance (consumer c_l) and each of the k centroids. The Euclidean distance is determined as follows:

$$dist\left(c_l, \mu_j\right) = \sqrt{\sum_{h=1}^{24}\left(c_{lh} - \mu_{jh}\right)^2}, \quad \forall j \in \overline{1,k} \qquad (1)$$

2) establish the allocation ρ_l of each consumer l to the cluster j based on the nearest centroid:

$$\rho_l := \arg\min dist\left(c_l, \mu_j\right)^2 \qquad (2)$$

Thus, each instance will be allocated to a single cluster. Let's denote by S_j the set of all instances allocated to each cluster (centroid) j, $\rho_l \in S_j$.

3) determine new centroids based on the mean value of the instances belonging to the corresponding clusters:

$$\mu_j = \frac{1}{\left|S_j\right|}\sum_{c_l \in S_j} c_l \qquad (3)$$

Steps 2 and 3 are repeated until the allocation of the instances do not change or a user-defined tolerance or maximum number of iterations is reached. The *k-means* can be seen as an optimization problem that minimizes the within-cluster sum of squared errors, known as cluster inertia.

As a result, the electricity consumers are grouped into k clusters representing seasonal load profiles. The allocation of each consumer to a load profile (ρ_l) will be considered as input for the ML algorithms.

A dimensionality reduction method is applied to select the most significant factors. For each region, meteorological factors (temperature, humidity, wind speed or cloud cover) have different influence and impact on the electricity consumption. Also, type of the day (weekdays or weekend days) and period of day (hours) influence to some extent the electricity consumption. To analyse their impact, we compared three feature selection algorithms: univariate selection, Recursive Feature Elimination (RFE) and Lasso Regularization (LassoR). For univariate selection, we use a feature selection method that computes the F score between each input factor and the electricity consumption. A small value will mean that the input factor has a minor influence on the consumption, whereas a big value will indicate a strong dependency; thus, the input factor has a major influence on the consumption. The method computes the scores for each input factor and automatically retains the most important factors with the highest scores selected as predictors (input parameters) for the ML algorithms. For the most important factors, we also calculate the Pearson correlation coefficient to validate the F scores. RFE uses the accuracy to rank the input factors considering their importance on the training model (1 being the most important) and recursively removes unimportant factors to build the model on the remaining attributes. It also provides the support of each factor as a Boolean value, *True* being a relevant factor and *False* being an irrelevant one. LassoR is an embedded method that iteratively extracts the factors with the highest impact on the training process at each iteration. If the factor is not important, then its coefficient (contribution) is set to zero. Hence, only the factors with non-zero coefficients may be kept for the model. Then, the rankings provided by these three algorithms are compared and the most important *fk* factors are selected considering their common highest scores. By applying the feature selection method, the number of input parameters are reduced, and the complexity of the ML algorithms is decreased, thus the STLF is performing faster and with less computational resources.

C. STLF WITH ML ALGORITHMS

1) DESCRIPTION OF VARIABLES AND PARAMETERS

The first four ML algorithms are developed on the FF-ANN architecture with backward propagation of errors using several options for optimizing the squared error function of the real output and the predicted values. The input parameters (X) are composed by m elements as follows: the most important *fk* meteorological factors, type of the day, hour and the allocation of the electricity consumers to one of the k clusters. The output of the STLF is represented by the electricity consumption (Y) for the next 24 hours. The FF-ANN architecture consists in three layers: the input layer (X) with m elements, one hidden layer (H) with p neurons and the output layer (Y) with a single element (electricity consumption). The output Y is determined based on the input X using the approximation function:

$$Y = f\,(X) \qquad (4)$$

The network topology is given in Figure 2 having only one hidden layer for faster training and performance.

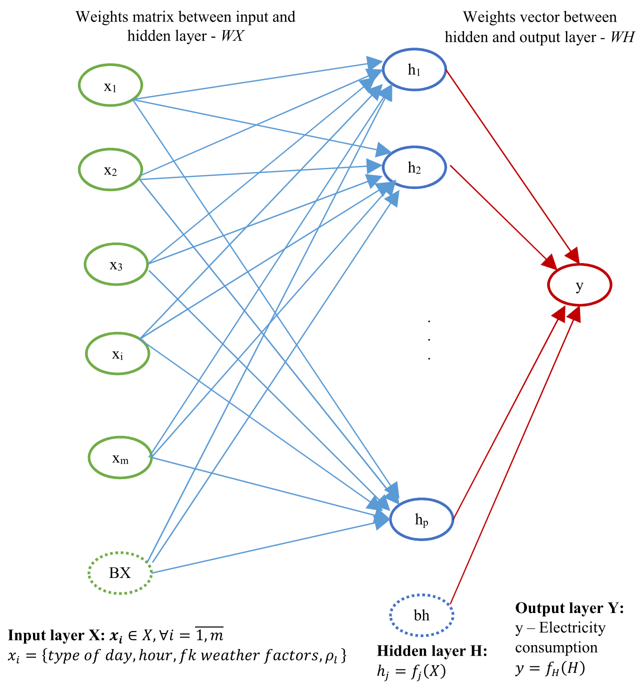
Weights matrix between input and hidden layer - WX
Weights vector between hidden and output layer - WH
x1
x2
x3
xi
xm
BX
h1
h2
hp
bh
y
Input layer X: $x_i \in X, \forall i = \overline{1,m}$
$x_i = \{type\ of\ day, hour, fk\ weather\ factors, \rho_l\}$
Hidden layer H: $h_j = f_j(X)$
Output layer Y: y – Electricity consumption $y = f_H(H)$

FIGURE 2. FF-ANN configuration.

Based on this topology, each element h_j of the hidden layer H is determined by the activation functions $f_j\,(X)$ as follows:

$$h_j = f_j(X) = f_j\left(bx_j + \sum_{i=1}^{m} wx_{ji}x_i\right), \quad \forall j = \overline{1,p} \tag{5}$$

In this case, the output Y is determined using the activation function $f_H(X)$ as follows:

$$Y = f_H(H) = f_H(bh + \sum_{j=1}^{p} wh_j h_j) \tag{6}$$

In order to simplify the notation and to increase the clarity of the algorithms, the weights and biases (WX, WH, BX, H, bh) are stored into a single in-memory variable W as a data frame. The elements $w_j \in W, j = \overline{1,p}$ are represented by $\{wx_{j.}, wh_j, bx_j, h_j, bh\}$. The algorithms automatically receive (from step 2.2) the size m of the input vector X and the number of hidden neurons p as parameters, and dynamically allocate the W variable.

2) TRAINING METHODS

The FF-ANN algorithms are using a supervised learning and attempt to minimize the errors between the actual values of the electricity consumption ($\hat{Y}$) and the output (Y) predicted with FF-ANN. The error function (E) for each pair ($\hat{Y}_q, X_q$) of the training set Q is adapted from [65]:

$$E = \frac{1}{2}(\hat{Y} - Y)^2 = \frac{1}{2Q}\sum_{q=1}^{Q} err_q^2 = \frac{1}{2Q}\sum_{q=1}^{Q}(\hat{Y}_q - Y_q)^2 \tag{7}$$

In order to find the minimum E, the principle of gradient descend method is used in which the weights ($wx_{ij}, wh_j, \forall i = \overline{1,n}\ \forall j = \overline{1,p}$) and biases ($bx_j, bh, \forall j = \overline{1,p}$) are adjusted at each learning step (iteration) considering the following equation:

$$\mathrm{w}_j^{t+1} = \mathrm{w}_j^t + \mathrm{lr}^t\mathrm{V}^t \tag{8}$$

However, for faster convergence and acceleration of error minimization, more efficient methods for updating the weights and biases are considered by introducing additional elements:

1) Algorithm MOMENTUM implements the Momentum method that accelerates the minimization of the error function in the downward direction and reduces the oscillations by adding the value updated at the previous step (t) to the current step ($t + 1$):

$$w_j^{t+1} = w_j^t - \eta w_j^{t-1} - lr\nabla E^t\left(w_j^t\right), \quad \forall j = 1,..,p \tag{9}$$

2) Algorithm NESTEROV implements the Nesterov method that reduces the convergence rate by adjusting the current values based on the gradient of the previous two iterations (t and t-1):

$$w_j^{t+1} = (1-\delta_t)\left(w_j^t - lr\nabla E^t\left(w_j^t\right)\right) + \delta_t\left(w_j^{t-1} - lr\nabla E^{t-1}\left(w_j^{t-1}\right)\right) \tag{10}$$

where δ_t starts with 1 and is updated iteratively based on another variable λ_t:

$$\delta_t = \frac{1-\lambda_t}{\lambda_{t+1}} \tag{11}$$

$$\lambda_t = \frac{1+\sqrt{1+4\lambda_{t-1}^2}}{2}; \quad \lambda_0 = 0; \tag{12}$$

The main advantage of the Nesterov method is that it obtains the lower bound of the convergence rate of order $1/t$^2, while the gradient descent method has a rate of convergence of order $1/t$ after t iterations.

3) Apart from the existing training methods, as a novelty, we propose the algorithm NEST_BCKTR that introduces the backtracking for adjusting lr^t in classic Nesterov method, keeping the λ_t constant as in Momentum method. For adjusting the learning rate, we implement Armijo-Goldstein method. This represents a line search method and its advantage is that it provides an optimum step to move along the line search (minimization of the network error). Initially, lr starts with 0.5 and gradually, at every iteration, is adjusted with a shrinking rate r, where $r \in (0, 1)$:

```
$lr^1 := 0.5$;
LOOP
$lr^{t+1} := r \times lr^t$;
UNTIL $\mathrm{E}^{t+1} - \mathrm{E}^t < lr^t \times \propto \times V^t$;
```

where $\propto$ is a constant between 0 and 0.5.

The weights are updated based on the following equation:

$$w_j^{t+1} = (1-\eta)\left(w_j^t - lr^t\nabla E^t\left(w_j^t\right)\right) + \eta\left(w_j^{t-1} - lr^{t-1}\nabla E^{t-1}\left(w_j^{t-1}\right)\right) \tag{13}$$

4) Considering the well-known performance for time-series in general and load forecast in particular [40], NARX algorithm is chosen to compare the accuracy of the proposed FF-ANN. Therefore, we implement NARX that uses nonlinear autoregressive network with exogenous variables to determine the electricity consumption ($\hat{Y}$) at step ($t + 1$) based on previous y_d past output and x_d past exogenous variables.

$$\hat{Y}(t+1) = f(Y(t), Y(t-1), \ldots, Y(t-y_d), X(t), X(t-1), \ldots, X(t-x_d)) \tag{14}$$

The most significant weather factors, cluster number, type of the day and hour are considered as exogenous variables. In our case, the previous three steps are considered for both y_d and x_d parameters.

The accuracy of the proposed FF-ANN algorithm is also compared with more competitive ML algorithms, i.e. DNN, GTB and RF.

5) DNN implements a sequential deep learning algorithm with a similar representation with FF-ANN, but with a higher number of hidden layers and more complex training methods. For comparison, we select Stochastic Gradient Descent (SGD) and ADAM as training methods. ADAM represents

an optimization algorithm for the first-order gradient using a stochastic objective function and proves its efficiency in terms of the computational resources [66]. In the simulation section, we test several configurations for DNN, i.e. different number of hidden layers, number of neurons of the hidden layers and activation functions (e.g. linear, rectified linear, SoftMax, hyperbolic tangent and sigmoid).

6) GTB implements the boosting ensemble algorithm that uses many predictors trained sequentially to minimize the network errors. GTB is especially applied for classification and regression problems and uses a weak prediction algorithm considering fixed size decision trees. At each iteration, GTB fits the current decision tree to pseudo-residuals attempting to minimize the errors of its predecessor using gradient descent method. The final prediction is obtained by computing the residuals predicted by all trees adjusted by the learning rate [67]. In the simulation section, we test different number of predictors with variable depth.

7) RF applies the bootstrap aggregation known as bagging that trains each decision tree of the ensemble algorithm with a subset randomly extracted from the training set. The final prediction is obtained by combining classifiers into two ways: averaging their probabilistic prediction or considering their voting for a single class [68].

3) SETTING UP THE PARAMETERS OF THE ML ALGORITHMS

Since the ML algorithms are sensitive to the data values and ranges, the data set is standardized using Median – MAD method described in [69]. This is proven to be a robust technique since it does not transform the values into a common numerical range. Also, this method is not sensitive to outliers or extreme values of the distribution. The standardized input values are transformed accordingly to the following equation:

$$x_i{}' = \frac{x_{i-}med_i}{MAD_i} \tag{15}$$

MAD_i is determined as follows:

$$\mathrm{MAD_i} = \mathrm{median}_{q=1}^{Q} \left| \mathrm{x_i^q} - \mathrm{med_i} \right| \tag{16}$$

Also, the training and validation steps of the algorithms require distinct sets of data, randomly selected from the whole data set, but with representative records. For this purpose, the data set is split into 80% training records and 20% validation records. The initialization step of the algorithms requires other important parameters, such as: the number of training epochs (*p_n_epoch*) and also the accepted error (*p_eps*) for stopping the training step.

These data pre-processing steps are implemented in Python as sub-routines developed as part of a single package used for obtaining the forecast (STLF_PACK package) shown in Figure 3.

4) CHOOSING THE BEST FORECASTING ALGORITHM

In order to automatically choose the BPA for STLF, the following performance indicators are evaluated:

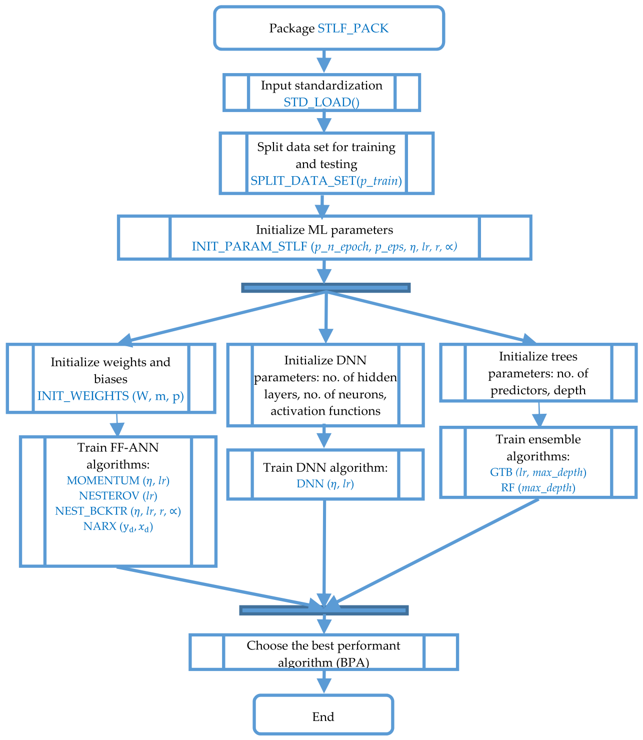
Package STLF_PACK
Input standardization STD_LOAD()
Split data set for training and testing SPLIT_DATA_SET(p_train)
Initialize ML parameters INIT_PARAM_STLF (p_n_epoch, p_eps, η, lr, r, ∝)
Initialize weights and biases INIT_WEIGHTS (W, m, p)
Initialize DNN parameters: no. of hidden layers, no. of neurons, activation functions
Initialize trees parameters: no. of predictors, depth
Train FF-ANN algorithms: MOMENTUM (η, lr) NESTEROV (lr) NEST_BCKTR (η, lr, r, ∝) NARX (y_d, x_d)
Train DNN algorithm: DNN (η, lr)
Train ensemble algorithms: GTB (lr, max_depth) RF (max_depth)
Choose the best performant algorithm (BPA)
End

FIGURE 3. **The flowchart of the training steps of the ML algorithms.**

- Error values in terms of average of the absolute error ($AVG\,(|E|)$), minimum absolute error ($MIN\,(|E|)$) and maximum absolute error ($MAX\,(|E|)$);
- Correlation coefficient between the actual consumption ($\hat{Y}$) and the forecasted consumption (Y);
- RMSE determined as follows:

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{q=1}^{Q}\left(\hat{Y}_q - Y_q\right)^2}{Q}} \tag{17}$$

The results are ordered first by RMSE and then by $AVG\,(|E|)$, $MIN\,(|E|)$ and $MAX\,(|E|)$. Then, the BPA is selected to provide the forecast of the electricity consumption for the next 24 hours. The flowchart of the training and validation steps of the above ML algorithms is synthetized in Figure 3.

Considering the above flowchart, we train the ML algorithms on a data set described in section 3 and compare the results in terms of accuracy for training and validation steps.

5) INCREMENTAL VALIDATION AND TESTING

In the previous steps, we have described the training and validation processes for the initial configuration of the ML algorithms. But for applying the algorithms in a testing or real environment, for day-ahead load forecasting, they must permanently adjust their configurations considering the most recent selected inputs. This process requires incremental validation considering the last period. Thus, the data set used for training and validation must be permanently updated by introducing the records measured in the last *d* days and removing the records measured in the first *d* days of the previous interval.

For testing, to provide the day-ahead load forecasting, the BPA is selected based on the accuracy of the last 24 hours forecasts obtained with all ML algorithms. In this case, the actual consumption values are available, therefore the performance indicators can be determined for each ML algorithm. The flowchart of the STLF in the testing environment is presented in Figure 4.

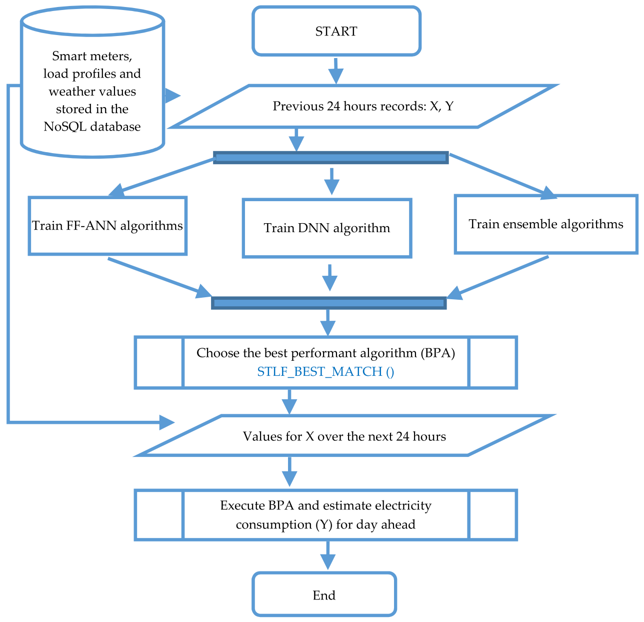
START
Smart meters, load profiles and weather values stored in the NoSQL database
Previous 24 hours records: X, Y
Train FF-ANN algorithms
Train DNN algorithm
Train ensemble algorithms
Choose the best performant algorithm (BPA)
STLF_BEST_MATCH ()
Values for X over the next 24 hours
Execute BPA and estimate electricity consumption (Y) for day ahead
End

FIGURE 4. **The flowchart of the ML algorithms in testing environment.**

Section 3 describes the data set, simulations and results obtained with the previously depicted methodology.

III. SIMULATIONS AND RESULTS

A. SMART METERING DATA SOURCE DESCRIPTION AND PRE-PROCESSING

The data set used in simulations consists in raw files recorded by smart meters representing 15 by 15 minutes electricity consumption values from mid of 2014 up to December 2015 (around 7.25 mil. of records) and minute by minute electricity consumption values for 2016 (around 57.46 mil. of records).

Smart meters were installed during 2014, so that some records are missing or inconsistent; therefore, we consider for training and validation the 2015 data set (over 6 mil. of records), keeping the 2016 data set for incremental validation and testing of the ML algorithms.

Also, hourly weather data for the entire period (such as: temperature, apparent temperature, humidity, visibility, pressure, wind speed, wind bearing, cloud cover, precipitation intensity, precipitation probability, dew point) is available for the simulations. The recorded data belongs to a residential building with 114 single-family apartments located in New England [70]. This region has a humid continental climate, long winters, cold, and heavy snow. The summer months are moderately warm, though summer is rather short, and rainfall is usually spread through the year.

For pre-processing, the ETL process transforms, validates and loads the files collected from smart meters and weather stations into MongoDB. The files regarding the electricity consumption contain the following attributes: timestamp, apartment id, active power (kWh) and status of the sensor. If the status is online, but the load value is out of the boundaries, then corrections are made by adjusting the load value with the average of the previous reading values. The files from the weather stations contain hourly readings of the following parameters: dew point, apparent temperature, temperature, humidity, visibility, precipitation intensity, precipitation probability, cloud cover, icon (sky-description), wind bearing, wind speed and atmospheric pressure. The descriptive variables are transformed into numerical values using encoders. The weather readings are transformed from hourly to 15 minutes data values using interpolation. Therefore, the electricity consumption is correlated with the meteorological readings and can be used as input for the ML algorithms.

B. CLUSTERING AND DIMENSIONALITY REDUCTION

To determine the load profiles, an initial analysis of the data set is performed, noticing that in most of the cases, the morning peak is at 8-9, the evening peak at 20-21, the night peak at 3 and the day off-peak at 15 as represented in Figure 5.

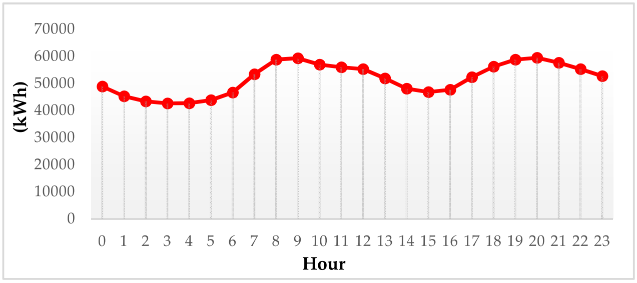

FIGURE 5. **Total hourly load curve for 2015 data set.**

However, there are significant differences regarding the individual consumption behaviour of the inhabitants of the 114 apartments in terms of electricity consumption. For instance, some of them may reach the morning/evening peak, while others may reach the night/day off-peak at the same hour. Therefore, splitting apartments into clusters is crucial for STLF.

The contribution of the apartments to the morning and evening peaks is depicted in Figure 6. It can be noticed that each apartment has a different contribution to the peaks. For instance, apartment no. 28 contributes with more than 2.2%, while apartment no. 30 contributes with less than 0.25% to the consumption peaks. This could provide valuable information to the designers of the time-of-use tariffs as they could impose higher rates based on contribution to the consumption peak.

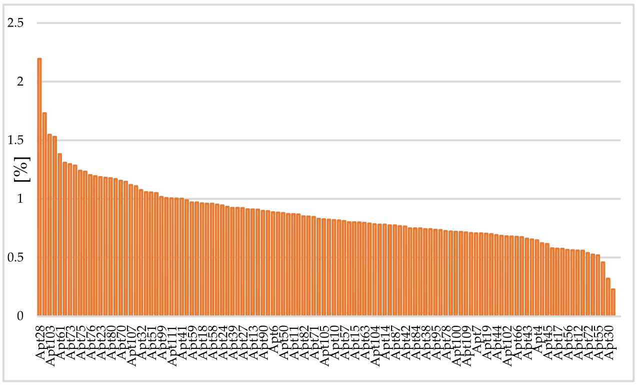

FIGURE 6. Load peaks contribution of the 114 apartments for 2015 data set.

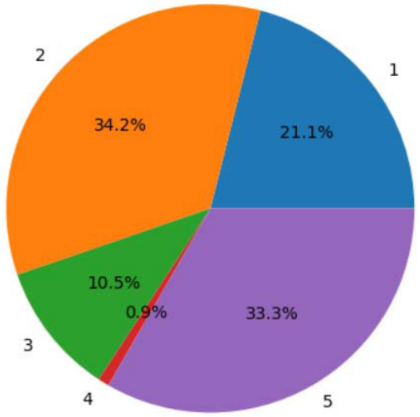

FIGURE 8. Members' distributions among the clusters.

After hourly aggregating the electricity consumption over each season for the entire data set, we cluster the 114 apartments to identify more similarities among the electricity consumers in terms of the hourly consumption. In order to determine the appropriate number of clusters, we perform the Elbow analysis (Figure 7) that indicates that the recommended value is 4 or 5.

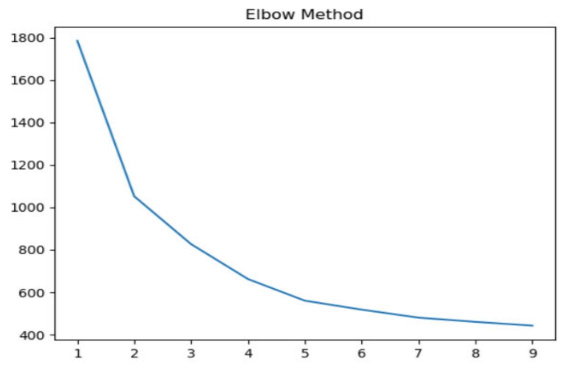

FIGURE 7. Elbow analysis indicating the recommended number of clusters.

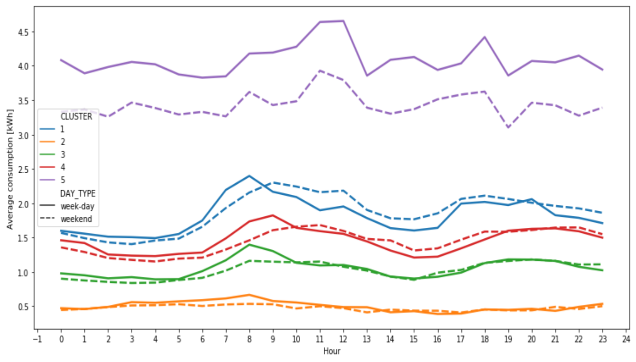

FIGURE 9. Load profiles for weekdays and weekend days determined for the entire year.

Therefore, 5 clusters with *k-means* algorithm are determined, considering the Euclidian distance. The size of each cluster is shown in Figure 8.

As described in the proposed methodology, the forecast is performed for each cluster, taking into account the type of the day, considering weekdays (from Monday to Friday) and weekend (Saturdays and Sundays) due to their specificities (i.e. on weekdays peaks, load are slightly higher, morning peak are earlier compared with the weekend peaks) as in Figure 9.

After this step completes, each apartment is assigned to a specific cluster and its allocation (ρ_l) can be used as input for the ML algorithms.

To automatically configure the ML algorithms input as vector X, the dimensionality reduction is applied to select the most important fk factors that influence the electricity consumption. As presented in the methodology, we compared three feature selection methods, all of them implemented in the scikit-learn Python package: SelectKBest with univariate linear regression test (*f_regression*), RFE and LassoCV for Lasso regularization. Table 2 depicts the rankings of each meteorological factor determined with SelectKBest (F scores), RFE (ranking and support) and LassoR coefficient.

TABLE 2. The influence of the weather parameters over the electricity consumption.

Feature_name	F score	RFE Ranking	RFE Support	LassoR coefficient
TEMPERATURE	2269.784283	1	True	0.8673
APPARENT_ TEMPERATURE	1991.448546	1	True	0.7589
DEWPOINT	1535.583203	1	True	0.5704
PRESSURE	293.164369	1	True	-0.1180
PRECIPPROBABILITY	195.692819	1	True	0.0003
WIND_SPEED	0.075695	1	True	-0.0047
CLOUD_COVER	205.839159	2	False	-0.0001
ICON	157.832394	6	False	0
PRECIPINTENSITY	78.662298	3	False	0
HUMIDITY	58.157512	5	False	0
VISIBILITY	30.982260	4	False	0
WIND_BEARING	1.042299	7	False	0

Analysing the 2015 data set by applying the Pearson correlation coefficient, we notice a very strong inverse correlation between the electricity consumption and the apparent temperature as in Figure 10 and relatively weak correlation between load and the rest of the weather parameters.

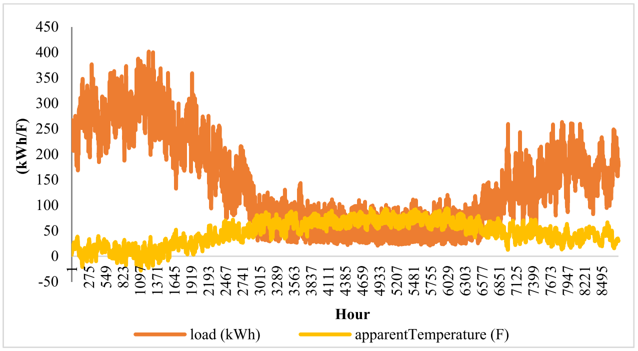

FIGURE 10. Hourly load and apparent temperature for 2015.

As it can be noticed, the most important weather factors, common to all methods are temperature, apparent temperature and dew point. Therefore, we set *fk* = 3 and these meteorological factors are selected as predictors for the ML algorithms.

The following input parameters are passed to the ML algorithms: X_1 – cluster number ($X_1 \in \{1, 2, 3, 4, 5\}$), X_2 – type of the day (day code, $X_2 \in \{1, 2\}$), X_3 - apparent temperature, X_4 – dew point, X_5 – temperature, X_6 – hour. The output Y is the electricity consumption.

C. STLF PERFORMANCE

1) INITIAL TRAINING AND VALIDATION RESULTS

First, we perform the input standardization to transform the input based on Median – MAD method. Then, the data set is split for training (80% of the records) and for validation (20% of the records).

For FF-ANN algorithms, similar neural network configuration is considered in order to compare their accuracy. After several tests, the best results are obtained with 100 training epochs (*p_n_epoch*), 40 neurons on the hidden layer (*p*), 0.01 for the initial learning rate and 0.1 for the accepted error (*p_eps*). The activation function is the rectified linear unit (*relu*). Similar results in terms of accuracy are obtained with hyperbolic tangent (*tanh*) and 64 neurons on the hidden layer, but with a slightly higher computational time for training. The weights and biases are randomly initialized with values between 0 and 0.5. For the DNN algorithm, we start with one input layer and 6 hidden layers with 32 neurons using *relu* as activation function and SGD as training method. After several tests, comparing the results in terms of accuracy and computational time, we set the final configuration to 3 hidden layers with 64 neurons, *tanh* activation function and ADAM as training method. Comparable results in terms of accuracy are also recorded with 2 hidden layers with lower computational time, meaning that the DNN developed for load forecast does not actually require a deep architecture. It also indicates that DNN shows limited advantages compared with FF-ANN.

For the GTB and RF algorithms, we test several configurations and finally we set the number of predictors to 60 and *max_depth* to 15. A lower value for *max_depth* reduces the computational time but decreases the accuracy.

The forecast time-horizon is 24-hour ahead at a 15-minute resolution and lead time is 1 hour. Update frequency is hourly based assuring a refresh of the results close to the real-time operation.

The ML algorithms are executed simultaneously and their results are first compared in terms of accuracy. For comparison, we train the algorithms without considering the clusters and then repeat the training on each cluster. Table 3 synthetises the results obtained for the entire data set (without clusters) and Table 4 synthetises the average results obtained for the 5 clusters.

TABLE 3. STLF performance for training and validation without clusters, 2015 data set.

	MOMENTUM	NESTEROV	NEST_BCKTR	NARX	DNN	GTB	RF
$AVG(\|E\|)$	4.72	4.56	4.3	5.98	4.27	**4.01**	4.11
$MIN(\|E\|)$	0.23	0.2	0.17	0.45	0.24	**0.14**	0.09
$MAX(\|E\|)$	16.54	14.69	14.33	18.11	**14.31**	14.79	15.25
RMSE	6.11	5.92	**5.72**	7.32	5.86	5.95	6.02
R	0.9596	0.9603	**0.9616**	0.9328	0.9613	0.9514	0.9567

TABLE 4. STLF performance for training and validation with 5 clusters, 2015 data set.

	MOMENTUM	NESTEROV	NEST_BCKTR	NARX	DNN	GTB	RF
$AVG(\|E\|)$	2.04	1.87	1.46	3.52	1.85	**1.23**	1.36
$MIN(\|E\|)$	0.0032	0.0035	**0.0012**	0.0823	0.0041	0.0021	0.0023
$MAX(\|E\|)$	6.45	5.73	**5.34**	9.52	5.96	7.35	7.81
RMSE	2.65	2.50	**1.93**	4.02	2.78	3.58	3.88
R	0.9896	0.9905	**0.9935**	0.9628	0.9921	0.9845	0.9869

Then, we analyse the performance recorded in the training and validation phase with the ML algorithms applied for each cluster by comparing the RMSE (Table 5).

TABLE 5. STLF performance for training and validation RMSE performance for each cluster, 2015 data set.

	MOMENTUM	NESTEROV	NEST_BCKTR	NARX	DNN	GTB	RF
C1	2.98	2.76	**1.82**	3.71	2.73	3.42	3.57
C2	2.56	2.45	2.15	3.21	2.19	**1.98**	2.03
C3	2.76	2.73	**1.69**	3.90	2.81	1.87	1.92
C4	2.62	2.32	**2.18**	3.97	2.45	2.35	2.39
C5	2.51	2.44	**1.89**	4.36	2.41	3.69	3.98

In the training phase, the FF-ANN algorithms (excepting NARX) and DNN provide very good accuracy with similar results in terms of RMSE for all clusters, independently of their distribution (size) or load profiles. Comparing the results of the ensemble algorithms and analysing the load profiles for each cluster, it can be noticed that RF and GTB record a very good accuracy on clusters no. 2 and 3 that have flatter curves, without significant peaks or off-peaks. On the other hand, on clusters 1 and 5 with frequent peaks and off-peaks, these algorithms have similar results with NARX which provides the lowest accuracy. Regarding the computational resources, we noticed a very high CPU activity and a significant increase in the computational time for training the DNN.

2) INCREMENTAL VALIDATION AND TESTING RESULTS

As mentioned before, the 2016 data set is preserved only for incremental validation and testing. We set $d = 15$ days and run the ML algorithms to select the BPA. After this period, the weights and biases are updated, and the data set used for the next validation process is also updated. Also, clustering technique is applied for each season to include the weather influence on the electricity consumption.

For testing, we consider twelve scenarios involving the most populated clusters (1, 2 and 5) and two types of the days: winter and summer weekdays (19-Jan-2016 and 7-Aug-2016) and winter and summer weekend days (24-Jan-2016 and 23-Aug-2016). The first scenario considers cluster 2 on a winter weekday (19-Jan-2016) and the algorithm with the best performance over the last 24 hours is automatically selected to forecast the electricity consumption for the next day. The performance indicators for this scenario are provided in Table 6.

TABLE 6. STLF performance at the testing phase: Cluster 2, 19-Jan-2016, winter weekday.

	MOMENTUM	NESTEROV	NEST_BCKTR	NARX	DNN	GTB	RF
$AVG(\|E\|)$	2.23	1.48	**1.24**	3.48	1.52	2.56	2.79
$MIN(\|E\|)$	0.47	0.52	0.16	0.91	0.38	**0.06**	0.56
$MAX(\|E\|)$	4.02	2.96	**2.10**	7.32	2.23	6.89	7.12
RMSE	1.84	1.35	**1.25**	3.05	1.49	3.15	3.29
R	0.9858	0.9928	**0.9934**	0.9772	0.9902	0.9714	0.9688

Even though with GTB and RF we obtain very good results at the training stage on cluster 2 and we expected to outperform the other ML algorithms at the testing stage, the best performance is recorded with NEST_BCKTR, so this algorithm provides the forecast for the next day. DNN has similar results with the FF-ANN that uses classic Nesterov as training method. The results obtained by NEST_BCKTR are displayed in Figure 11.

The second scenario considers a winter weekend day (24-Jan-2016) for cluster 2. In this case, NESTEROV algorithm records the best performance (RMSE = 1.32) comparing to the other six algorithms on the day before (that is also a weekend day), so it provides the forecast for next day.

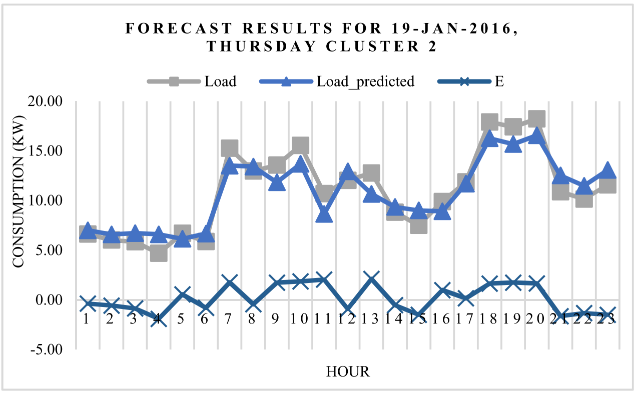

FIGURE 11. Load forecast for cluster 2 for a winter weekday.

Similar results are recorded with DNN (RMSE = 1.38), followed by NEST_BCKTR (RMSE = 1.39) and MOMENTUM (RMSE = 2.15). The lowest accuracy is obtained with NARX and the ensemble algorithms with RMSE between 3.17 and 4.7. The forecast results obtained with NESTEROV are displayed in Figure 12.

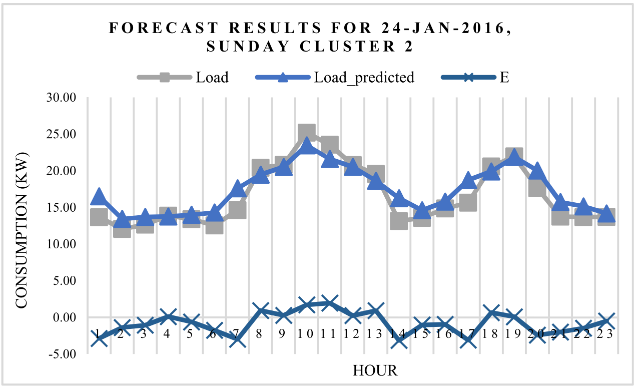

FIGURE 12. Load forecast for cluster 2 for a winter weekend day.

For the third scenario, we forecast the electricity consumption of cluster 5 on 19-Jan-2016 with NEST_BCKTR, automatically selected by the process flow, having RMSE = 2.12. Very good results are also recorded with NESTEROV (RMSE = 2.29) and DNN (RMSE = 2.49) followed by MOMENTUM with RMSE = 2.61. The lowest accuracy is obtained with NARX and the ensemble methods were RMSE is between 3.4 and 5.01. The forecast results obtained with NEST_BCKTR are presented in Figure 13.

In the fourth scenario, for cluster 5, we forecast the electricity consumption for a winter weekend day (24-Jan-2016). For the day before, the best performance is recorded with NEST_BCKTR with RMSE = 2.01, followed by NESTEROV (RMSE = 2.24) and DNN (RMSE = 2.48). The forecast results obtained with NEST_BCKTR are presented in Figure 14.

In the fifth scenario, cluster 1 is also considered for simulations to validate the results for other clusters with

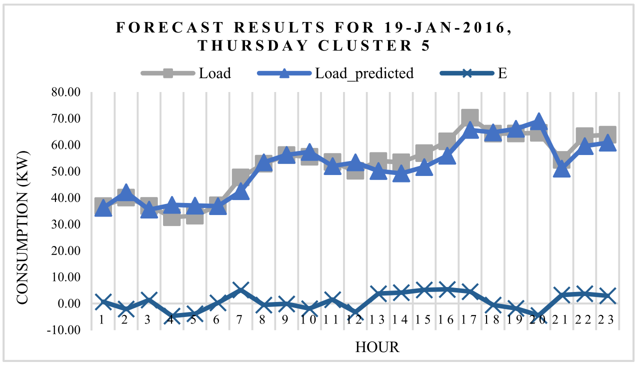

FIGURE 13. Load forecast for cluster 5 for a winter weekday.

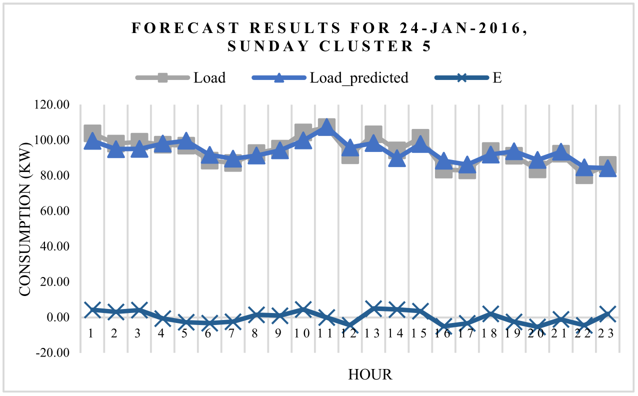

FIGURE 14. Load forecast for cluster 5 for a winter weekend day.

numerous members. The electricity consumption for cluster 1 on 19-Jan-2016 is forecasted with NEST_BCKTR having the best RMSE = 2.37. In this case, there are some minor spikes in the real load that are not very well fitted by the prediction as it can be observed in Figure 15 for hours 9 and 12.

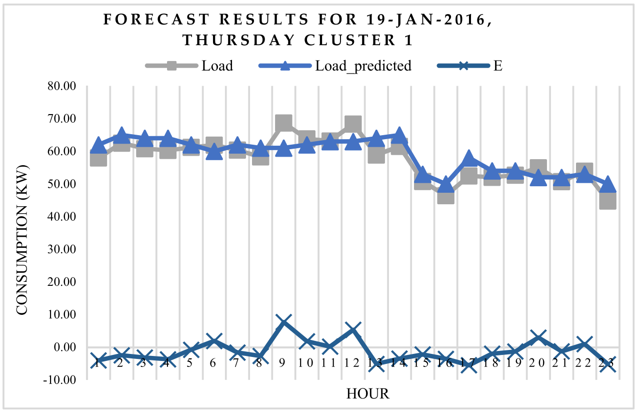

FIGURE 15. Load forecast for cluster 1 for a winter weekday.

In the sixth scenario, the forecast is also performed for winter weekend day (24-Jan-2016) for cluster 1. The BPA is NESTEROV with best RMSE = 2.27. The results are presented in Figure 16.

The results for winter simulations are centralized in Table 7.

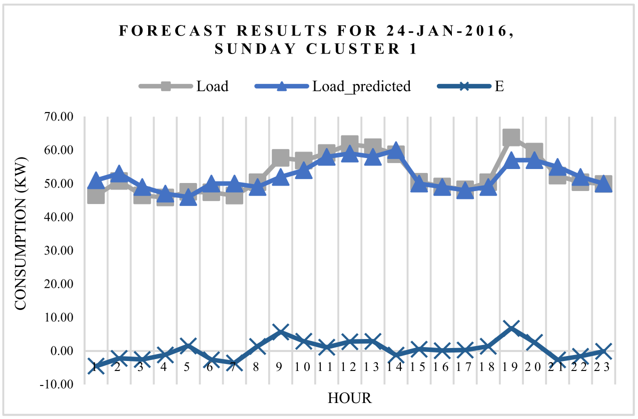

FIGURE 16. Load forecast for cluster 1 for a winter weekend day.

TABLE 7. STLF performance (RMSE) in the testing phase for clusters 1, 2 and 5 on winter days.

Cluster	MOMENTUM	NESTEROV	NEST_BCKTR	NARX	DNN	GTB	RF
			Winter weekday 19-Jan-2016				
C1	2.78	2.51	**2.37**	3.63	2.49	3.8	4.5
C2	1.84	1.35	**1.25**	3.05	1.49	3.15	3.29
C5	2.61	2.29	**2.12**	3.40	2.49	4.8	5.01
			Winter weekend day 24-Jan-2016				
C1	2.64	**2.27**	2.35	3.56	2.39	3.61	4.02
C2	2.15	**1.32**	1.39	3.17	1.38	4.1	4.7
C5	2.63	2.24	**2.01**	3.53	2.48	4.5	5.2

For all winter scenarios, the ML algorithms provide accurate forecast, either for weekdays or for weekend. In most cases, the NEST_BCKTR has the best performance in term of RMSE. In case of two winter weekend days, for clusters 1 and 2, the BPA is NESTEROV. Very good results are obtained with DNN in all scenarios, while the lowest accuracy was obtained with NARX and the two ensemble algorithms, namely GTB and RF.

In order to better evaluate the results, we also run the simulations for clusters 1, 2 and 5 for two summer days: 7-Aug-2016 (weekend day) and 23-Aug-2016 (weekday). The forecast performance of the ML algorithms is centralized in Tables 8 and 9.

TABLE 8. STLF performance (RMSE) in the testing phase. Clusters 1, 2 and 5 for summer weekend day (7-Aug-2016).

	MOMENTUM	NESTEROV	NEST_BCKTR	NARX	DNN	GTB	RF
C1	2.89	2.24	2.36	3.75	**2.22**	4.26	4.38
C2	2.65	2.03	**1.96**	3.15	2.05	2.69	2.87
C5	3.02	2.96	**2.52**	4.21	2.78	4.58	5.01

The best performance in case of summer weekend day for cluster 1 is obtained with DNN, while for clusters 2 and 5, NEST_BCKTR performed better. Also, NESTEROV recorded very similar results. As it can be

TABLE 9. STLF performance (RMSE) in the testing phase. Clusters 1, 2 and 5 for summer weekday (23-Aug-2016).

	MOMENTUM	NESTEROV	NEST_BCKTR	NARX	DNN	GTB	RF
C1	2.75	2.54	**2.19**	3.68	2.49	4.38	4.58
C2	2.16	1.85	**1.66**	3.28	2.01	2.06	2.19
C5	2.86	2.05	**1.84**	4.15	2.26	4.89	5.06

observed, the differences between these three methods are very small, so each method can be used to provide a good forecast for summer weekend days. In case of cluster 2, a good accuracy is obtained with DNN, followed by the ensemble algorithms, MOMENTUM and NARX.

For clusters 1, 2 and 5, in summer weekdays, NEST_BCKTR performs better than the other ML algorithms, recording smaller differences compared to DNN and NESTEROV and bigger difference compared to NARX. This proves that both algorithms based on Nesterov learning method are reliable for an accurate forecast in case of the considered data set. Even though Momentum method did not perform as well as Nesterov in these scenarios, it is possible that with different data sets to provide better accuracy since its results are relatively close to the Nesterov methods. DNN records a very good accuracy, similar with the algorithms based on Nesterov learning method. While for the cluster 2, the ensemble algorithms obtain good results, similar with DNN in terms of RMSE, for clusters 1 and 5 their results are similar with NARX, RF having the lowest accuracy.

Besides the accuracy of the ML algorithms, we compare the computational time, CPU activity and memory allocation during the training and testing stages. Thus, the lowest resources are required by the two Nesterov methods as both imply faster convergence than the other ML algorithms. Also, a small computational time and a low CPU activity, but an increased memory allocation are required by the ensemble algorithms. The highest resources are consumed by the DNN which requires almost 38 times more computational time than Nesterov algorithms. In this case, a very high CPU activity (around 96%) is also noticed. In the context of smart metering and Big Data, a small computational time and less resources required by the ML algorithms are extremely important in selecting the BPA. For the proposed algorithm, NEST_BCKTR, the computational time is reduced compared to the classical Nesterov method, since it uses a more efficient method for minimizing the network error by implementing a dynamical adjustment of the learning rate.

IV. CONCLUSION

In this paper, we propose an integrated methodology for STLF using ML algorithm in the Big Data context that mainly involves the processing of large volumes of data recorded at different time resolutions. Our approach dynamically determines the most important attributes as predictors considering their influence on the electricity consumption. Thus, the proposed methodology can be applied in different regions, automatically detecting the relevant meteorological factors that will be considered as input, reducing the time and resources for additional investigations related to the selection of attributes. Also, by selecting significant predictors, the ML algorithms configuration is reduced, and the performance is increased due to the noise filtering of uncorrelated attributes.

Another important aspect of the proposed methodology represents the clustering of the electricity consumers obtaining useful load profiles. By grouping the consumers into relevant clusters considering the consumption pattern similarities, the performance of STLF is improved, RMSE decreasing from 5.72 to 1.93.

Our approach in terms of STLF relies on seven ML algorithms (i.e. three FF-ANN algorithms, NARX, DNN, GTB and RF) that simultaneously run to select the BPA and estimate the electricity consumption for the next day. The FF-ANN algorithms are based on the enhanced gradient descend methods, two of them using existing methods, such as Momentum and Nesterov. Besides, we proposed and implemented a novel learning method by introducing a backtracking adjustment of the learning rate.

By selecting the ML algorithm with the best performance, a more accurate prediction is obtained, the results being incrementally validated. The algorithms are implemented in Python, trained, validated and tested on 2-year data sets; the results are compared in terms of RMSE, correlation coefficient R, average, minimum and maximum of absolute error.

The proposed algorithm, NEST_BCKTR, performs better than the other ML algorithms, revealing smaller RMSE differences compared to the classic Nesterov and DNN. For training the DNN, very high computational resources were required, almost 38 times more than Nesterov algorithms. This proves that both algorithms based on Nesterov learning method are reliable for an accurate and faster forecast in case of the considered data set. NARX and ensemble algorithms record lower performances in terms of accuracy. However, GTB and RF algorithms provide similar results in terms of accuracy as Nesterov and DNN algorithms for summer days in case of clusters 2 and 3 with smooth load profiles. Even though Momentum didn't perform as well as Nesterov and DNN in the analysed scenarios, it is possible that with different data sets to offer better accuracy since its accuracy is acceptable for all clusters.

Another interesting conclusion is referring the optimal DNN configuration that revealed a number of only 3 hidden layers (or even 2) showing limited advantages compared with FF-ANN.

A limitation of our approach is that the input data for temperature, apparent temperature and dew point are measured values, not estimations, so there is no error probability for the weather forecast. But, in a real environment, the weather data will be also an estimation. This aspect needs to be further investigated by introducing a probabilistic forecast regarding the weather conditions.

NOMENCLATURE

Symbol	Description
c_{lh}	The average hourly consumption of consumer l at hour h, $\forall l = \overline{1,nl}$, where nl represents the total number of electricity consumers in a specific region and $h = \overline{1,24}$
μ_j	The centroid of the cluster j, $\forall j \in \overline{1,k}$
ρ_l	The allocation ρ_l of each consumer l to the cluster j
S_j	The set of all instances allocated to each cluster (centroid) j, $\rho_l \in S_j$.
Y	Single element output vector representing the electricity consumption, $Y = y$
m	Number of independent variables
X	Input vector of the ANN $X = (x_1, \dots, x_i, \dots, x_m)'$, $\forall i = \overline{1,m}$
p	Number of neurons of the hidden layer
H	Vector of the hidden layer, $H = (h_1, \dots, h_j, \dots, h_p)'$, $\forall j = \overline{1,p}$
$f_j(X)$	Activation functions of the hidden layer, $\forall j \in \overline{1,p}$
BX	Bias vector for the input layer, $BX = (bx_1, \dots, bx_j, \dots, bx_p)'$, $\forall j = \overline{1,p}$
WX	Weights' matrix between the input and the hidden layer, $WX = \begin{bmatrix} wx_{11} & wx_{1i} & wx_{1m} \\ wx_{j1} & wx_{ji} & wx_{jm} \\ wx_{p1} & wx_{pi} & wx_{pm} \end{bmatrix}$, $\forall i = \overline{1,m}$ and $j = \overline{1,p}$
$f_H(X)$	Activation function of the output layer
bh	Bias for the hidden layer (a single value variable)
WH	Weight vector between the hidden and the output layer, $WH = (wh_1, \dots, wh_j, \dots, wh_p)'$, $\forall j = \overline{1,p}$
x_i'	Standardized elements of vector X, $\forall i \in \overline{1,m}$
Q	Total number of records of the data set
q	Record of the training set , $q \in \overline{1,Q}$
med_i	Median of element i, $\forall i \in \overline{1,m}$, $med_i = median_{q=1}^{Q} x_i^q$
MAD_i	Median absolute deviation of the element i, $\forall i \in \overline{1,m}$
$\hat{Y}$	Actual value of the electricity consumption
E	Error function
err_q	Error between the actual values of electricity consumption ($\hat{Y}$) and the output (Y) for each record $q \in \overline{1,Q}$
t	Current iteration (step)
lr^t	Learning rate at current iteration
V^t	Direction of the negative gradient of E at current iteration, $V^t = -\nabla E^t(w_j^t)$
η	Adjustment rate of the previous update for Momentum method
λ_t	Adjustment coefficient for Nesterov method
δ_t	Adjustment coefficient that starts with 1 and iteratively is updated based on λ_t
$\propto$	Constant between 0 and 0.5 used for adjustment of the learning rate
y_d	Past output considered for NARX method
x_d	Past input considered for NARX method for exogenous variables
d	Number of days considered for incremental validations
W	Data frame variable storing weights and biases of the FF-ANN, $w_j \in W, j = \overline{1,p}$
r	Shrinking rate for NEST_BCKTR algorithm, $r \in (0,1)$

REFERENCES

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

[2] W. Charytoniuk, M. S. Chen, and P. Van Olinda, "Nonparametric regression based short-term load forecasting," *IEEE Trans. Power Syst.*, vol. 13, no. 3, pp. 725–730, Aug. 1998.

[3] S. Ružić, A. Vučković, and N. Nikolić, "Weather sensitive method for short term load forecasting in electric power utility of Serbia," *IEEE Trans. Power Syst.*, vol. 18, no. 4, pp. 1581–1586, Nov. 2003.

[4] T. Hong, M. Gui, M. E. Baran, and H. L. Willis, "Modeling and forecasting hourly electric load by multiple linear regression with interactions," in *Proc. IEEE PES Gen. Meeting*, Jul. 2010, pp. 1–8.

[5] Y. Chen, P. B. Luh, C. Guan, Y. Zhao, L. D. Michel, M. A. Coolbeth, P. B. Friedland, and S. J. Rourke, "Short-term load forecasting: Similar day-based wavelet neural networks," *IEEE Trans. Power Syst.*, vol. 25, no. 1, pp. 322–330, Feb. 2010.

[6] G. Sudheer and A. Suseelatha, "Short term load forecasting using wavelet transform combined with Holt–Winters and weighted nearest neighbor models," *Int. J. Electr. Power Energy Syst.*, vol. 64, pp. 340–346, 2015.

[7] B. Dong, Z. Li, S. M. M. Rahman, and R. Vega, "A hybrid model approach for forecasting future residential electricity consumption," *Energy Build.*, vol. 117, pp. 341–351, 2016.

[8] W. Charytoniuk and M.-S. Chen, "Very short-term load forecasting using artificial neural networks," *IEEE Trans. Power Syst.*, vol. 15, no. 1, pp. 263–268, Feb. 2000.

[9] P. Mandal, T. Senjyu, N. Urasaki, and T. Funabashi, "A neural network based several-hour-ahead electric load forecasting using similar days approach," *Int. J. Electr. Power Energy Syst.*, vol. 28, no. 6, pp. 367–373, 2006.

[10] W. He, "Load forecasting via deep neural networks," *Procedia Comput. Sci.*, 2017.

[11] H. Shi, M. Xu, and R. Li, "Deep learning for household load forecasting—A novel pooling deep RNN," *IEEE Trans. Smart Grid*, vol. 9, no. 5, pp. 5271–5280, Sep. 2018.

[12] K. Amarasinghe, D. L. Marino, and M. Manic, "Deep neural networks for energy load forecasting," in *Proc. IEEE 26th Int. Symp. Ind. Electron.*, Jun. 2017, pp. 1483–1488.

[13] G. M. U. Din and A. K. Marnerides, "Short term power load forecasting using deep neural networks," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, 2017, pp. 1–5.

[14] M. Cai, M. Pipattanasomporn, and S. Rahman, "Day-ahead building-level load forecasts using deep learning vs. traditional time-series techniques," *Appl. Energy*, vol. 236, pp. 1078–1088, 2019.

[15] T. Hossen, A. S. Nair, R. A. Chinnathambi, and P. Ranganathan, "Residential load forecasting using deep neural networks (DNN)," in *Proc. North Amer. Power Symp. (NAPS)*, 2019, pp. 1–5.

[16] F. Fahiman, S. M. Erfani, S. Rajasegarar, M. Palaniswami, and C. Leckie, "Improving load forecasting based on deep learning and K-shape clustering," in *Proc. Int. Joint Conf. Neural Netw.*, 2017, pp. 4134–4141.

[17] S. Papadopoulos and I. Karakatsanis, "Short-term electricity load forecasting using time series and ensemble learning methods," in *Proc. IEEE Power Energy Conf. Illinois (PECI)*, Feb. 2015, pp. 1–6.

[18] S. Ahmadi, H. Bevrani, and H. Jannaty, "A fuzzy inference model for short-term load forecasting," in *Proc. 2nd Iranian Conf. Renew. Energy Distrib. Gener. (ICREDG)*, 2012, pp. 39–44.

[19] H. H. Çevik and M. Çunkaş, "Short-term load forecasting using fuzzy logic and ANFIS," *Neural Comput. Appl.*, vol. 26, no. 6, pp. 1355–1367, 2015.

[20] G. C. Liao, "An improved fuzzy neural networks approach for short-term electrical load forecasting," in *Proc. 8th Asian Control Conf.-Final Program (ASCC)*, 2011.

[21] D. Niu, Y. Wang, and D. D. Wu, "Power load forecasting using support vector machine and ant colony optimization," *Expert Syst. Appl.*, vol. 37, no. 3, pp. 2531–2539, 2010.

[22] A. Selakov, D. Cvijetinović, L. Milović, S. Mellon, and D. Bekut, "Hybrid PSO-SVM method for short-term load forecasting during periods with significant temperature variations in city of Burbank," *Appl. Soft Comput. J.*, vol. 16, pp. 80–88, 2014.

[23] R. Rahmani, R. Yusof, M. Seyedmahmoudian, and S. Mekhilef, "Hybrid technique of ant colony and particle swarm optimization for short term wind energy forecasting," *J. Wind Eng. Ind. Aerodyn.*, vol. 123, pp. 163–170, 2013.

[24] H. Chitsaz, H. Shaker, H. Zareipour, D. Wood, and N. Amjady, "Short-term electricity load forecasting of buildings in microgrids," *Energy Build.*, vol. 99, pp. 50–60, 2015.

[25] P. Lusis, K. R. Khalilpour, L. Andrew, and A. Liebman, "Short-term residential load forecasting: Impact of calendar effects and forecast granularity," *Appl. Energy*, vol. 205, pp. 654–669, 2017.

[26] M. Ghayekhloo, M. B. Menhaj, and M. Ghofrani, "A hybrid short-term load forecasting with a new data preprocessing framework," *Electr. Power Syst. Res.*, vol. 119, pp. 138–148, 2015.

[27] H. Nie, G. Liu, X. Liu, and Y. Wang, "Hybrid of ARIMA and SVMs for short-term load forecasting," *Energy Procedia*, vol. 16, pp. 1455–1460, 2012.

[28] A. Kavousi-Fard, H. Samet, and F. Marzbani, "A new hybrid modified firefly algorithm and support vector regression model for accurate short term load forecasting," *Expert Syst. Appl.*, vol. 41, no. 13, pp. 6047–6056, 2014.

[29] H. Hahn, S. Meyer-Nieberg, and S. Pickl, "Electric load forecasting methods: Tools for decision making," *Eur. J. Oper. Res.*, vol. 199, no. 3, pp. 902–907, 2009.

[30] M. De Felice and X. Yao, "Short-term load forecasting with neural network ensembles: A comparative study," *IEEE Comput. Intell. Mag.*, vol. 6, no. 3, pp. 47–56, Aug. 2011.

[31] E. Kyriakides and M. Polycarpou, "Short term electric load forecasting: A tutorial," in *Trends in Neural Computation* (Studies in Computational Intelligence). Berlin, Germany: Springer, 2006.

[32] H. X. Zhao and F. Magoulés, "A review on the prediction of building energy consumption," *Renew. Sustain. Energy Rev.*, vol. 16. no. 6, pp. 3586–3592, 2012.

[33] N. Amjady and F. Keynia, "A new neural network approach to short term load forecasting of electrical power systems," *Energies*, vol. 4, no. 3, pp. 488–503, 2011.

[34] A. P. A. da Silva and L. S. Moulin, "Confidence intervals for neural network based short-term load forecasting," *IEEE Trans. Power Syst.*, vol. 15, no. 4, pp. 1191–1196, Nov. 2000.

[35] M. Q. Raza and A. Khosravi, "A review on artificial intelligence based load demand forecasting techniques for smart grid and buildings," *Renew. Sustain. Energy Rev.*, vol. 50, pp. 1352–1372, 2015.

[36] H. Quan, D. Srinivasan, and A. Khosravi, "Uncertainty handling using neural network-based prediction intervals for electrical load forecasting," *Energy*, vol. 73, pp. 916–925, 2014.

[37] S. Vasudevan, "One step ahead load forecasting for smart-grid applications," M.S. thesis, Dept. Elect. Comput. Eng., Ohio State Univ., Columbus, OH, USA, 2011.

[38] R. A. Sevlian and R. Rajagopal, "A model for the effect of aggregation on short term load forecasting," in *Proc. IEEE Power Energy Soc. Gen. Meeting*, Jul. 2014, pp. 1–5.

[39] A. Baliyan, K. Gaurav, and S. K. Mishra, "A review of short term load forecasting using artificial neural network models," *Procedia Comput. Sci.*, vol. 48, pp. 121–125, 2015.

[40] Z. Yun, Z. Quan, S. Caixin, L. Shaolan, L. Yuming, and S. Yang, "RBF neural network and ANFIS-based short-term load forecasting approach in real-time price environment," *IEEE Trans. Power Syst.*, vol. 23, no. 3, pp. 853–858, Aug. 2008.

[41] L. Hernandez, C. Baladrón, J. M. Aguiar, B. Carro, A. J. Sanchez-Esguevillas, and J. Lloret, "Short-term load forecasting for microgrids based on artificial neural networks," *Energies*, vol. 6, no. 3, pp. 1385–1408, 2013.

[42] K.-H. Kim, H.-S. Youn, and Y.-C. Kang, "Short-term load forecasting for special days in anomalous load conditions using neural networks and fuzzy inference method," *IEEE Trans. Power Syst.*, vol. 15, no. 2, pp. 559–565, May 2000.

[43] S. Banik, M. Anwer, and A. F. M. K. Khan, "Predictive power of the daily Bangladeshi exchange rate series based on Markov model, neuro fuzzy model and conditional heteroskedastic model," in *Proc. 12th Int. Conf. Comput. Inf. Technol. (ICCIT)*, 2009, pp. 303–308.

[44] S. Singh, S. Hussain, and M. A. Bazaz, "Short term load forecasting using artificial neural network," in *Proc. 4th Int. Conf. Image Inf. Process. (ICIIP)*, 2018, pp. 233–238.

[45] O. Tanidir and O. B. Tör, "Accuracy of ann based day-ahead load forecasting in Turkish power system: Degrading and improving factors," *Neural Netw. World*, vol. 15, no. 4, pp. 443–456, 2015.

[46] I. I. Esener, T. Yüksel, and M. Kurban, "Short-term load forecasting without meteorological data using AI-based structures," *Turkish J. Electr. Eng. Comput. Sci.*, vol. 23, pp. 370–380, 2015.

[47] Y. Li, P. Guo, and X. Li, "Short-term load forecasting based on the analysis of user electricity behavior," *Algorithms*, vol. 9, no. 4, pp. 1–14, 2016.

[48] F. Rodrigues, C. Cardeira, and J. M. F. Calado, "The daily and hourly energy consumption and load forecasting using artificial neural network method: A case study using a set of 93 households in Portugal," *Energy Procedia*, vol. 62, pp. 220–229, 2014.

[49] P. Wang, B. Liu, and T. Hong, "Electric load forecasting with recency effect: A big data approach," *Int. J. Forecasting*, vol. 32, no. 3, pp. 585–597, 2016.

[50] I. P. Panapakidis, "Clustering based day-ahead and hour-ahead bus load forecasting models," *Int. J. Electr. Power Energy Syst.*, vol. 80, pp. 171–178, 2016.

[51] S.-V. Oprea, A. Pîrjan, G. Cărutasu, D.-M. Petrosanu, A. Bâra, J.-L. Stănică, and C. Coculescu, "Developing a mixed neural network approach to forecast the residential electricity consumption based on sensor recorded data," *Sensors*, vol. 18, no. 5, pp. 1–41, 2018.

[52] C. Fan, F. Xiao, and Y. Zhao, "A short-term building cooling load prediction method using deep learning algorithms," *Appl. Energy*, 2017.

[53] A. Almalaq and G. Edwards, "A review of deep learning methods applied on load forecasting," in *Proc. 16th IEEE Int. Conf. Mach. Learn. Appl. ICMLA*, Dec. 2017, pp. 511–516.

[54] D. L. Marino, K. Amarasinghe, and M. Manic, "Building energy load forecasting using deep neural networks," in *Proc. Ind. Electron. Conf. (IECON)*, 2016, pp. 7046–7051.

[55] W. Kong, Z. Y. Dong, Y. Jia, D. J. Hill, Y. Xu, and Y. Zhang, "Short-term residential load forecasting based on LSTM recurrent neural network," *IEEE Trans. Smart Grid*, vol. 10, no. 1, pp. 841–851, Jan. 2019.

[56] J. Zheng, C. Xu, Z. Zhang, and X. Li, "Electric load forecasting in smart grids using long-short-term-memory based recurrent neural network," in *Proc. 51st Annu. Conf. Inf. Sci. Syst. (CISS)*, 2017, pp. 1–6.

[57] Z. Zhao, W. Chen, X. Wu, P. C. V. Chen, and J. Liu, "LSTM network: A deep learning approach for short-term traffic forecast," *IET Image Process.*, vol. 11, no. 2, pp. 68–75, 2017.

[58] J. Bedi and D. Toshniwal, "Deep learning framework to forecast electricity demand," *Appl. Energy*, vol. 238, pp. 1312–1326, 2019.

[59] C. Tian, J. Ma, C. Zhang, and P. Zhan, "A deep neural network model for short-term load forecast based on long short-term memory network and convolutional neural network," *Energies*, vol. 11, no. 12, pp. 1–13, 2018.

[60] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 2, pp. 295–307, Feb. 2016.

[61] J. Heaton, *Introduction to Neural Networks for Java*, 2nd ed. London, U.K.: Heaton Research, 2005.

[62] K. Chen, K. Chen, Q. Wang, Z. He, J. Hu, and J. He, "Short-term load forecasting with deep residual networks," *IEEE Trans. Smart Grid*, vol. 10, no. 4, pp. 3943–3952, Jul. 2019.

[63] E. Mocanu, P. H. Nguyen, M. Gibescu, and W. L. Kling, "Deep learning for estimating building energy consumption," *Sustain. Energy, Grids Netw.*, vol. 6, pp. 91–99, 2016.

[64] S.-V. Oprea, B. G. Tudorica, A. Belciu, and I. Botha, "Internet of Things, challenges for demand side management," *Inf. Econ.*, vol. 21, no. 4, pp. 59–72, 2017.

[65] Wikipedia. *Backpropagation*. [Online]. Available: https://en.wikipedia.org/wiki/Backpropagation

[66] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," Cornell Univ., Ithaca, NY, USA, Tech. Rep., 2014. [Online]. Available: https://arxiv.org/abs/1412.6980v8

[67] T. G. Dietterich, G. Hao, and A. Ashenfelter, "Gradient tree boosting for training conditional random fields," *J. Mach. Learn. Res.*, vol. 9, pp. 2113–2139, 2008.

[68] L. Breiman, "Randomforest2001," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[69] P. J. Rousseeuw and C. Croux, "Alternatives to the median absolute deviation," *J. Amer. Stat. Assoc.*, vol. 88, no. 424, pp. 1273–1283, 1993.

[70] UMassTraceRepository. *Smart* Data Set for Sustainability*. Accessed: May 2019. [Online]. Available: http://traces.cs.umass.edu/index.php/Smart/ Smart