Analysis and enhanced prediction of the Spanish Electricity Network through Big Data and Machine Learning techniques

M.C. Pegalajar[a] *, L. G. B. Ruiz[a,b,*], M.P. Cuéllar[a], R. Rueda[a]

mcarmen@decsai.ugr.es*, bacaruiz@ugr.es*, manupc@decsai.ugr.es, ramonrd@ugr.es

[a] *Department of Computer Science and Artificial Intelligence, University of Granada, Spain.*

[b] *Department of Software Engineering, University of Granada, Spain.*

ABSTRACT

Electricity demand is shown to steadily increase in the last few years, and it is one of the key aspects of living standards and quantifying welfare effects. However, the irregularity of electricity demand is one of the main problems in this field. Therefore, it is important to accurately anticipate future expenditures in order to optimize energy generation and to avoid unexpected wastes. As a result, we developed Machine Learning models to predict electricity demand. In particular, our study has been performed using data of the Spanish Electricity Network from 2007 to 2019. To this end, we propose the implementation of a set of Machine Learning techniques using various frameworks. In particular, we implemented six different prediction models: Linear Regression, Regression Trees, Gradient Boosting Regression, Random Forests, Multi-layer Perceptron, and three types of recurrent neural networks. Our experimentation shows promising results in all cases, since our models provides better prediction than the one estimated by the Spanish Electricity Network with an improvement of 12% in the worst case and up to 37% for the best predictor, which turned out to be the Gated Recurrent Unit neural network.

Keywords:
Deep learning
Electricity demand
Machine Learning
Time-series forecasting

1. Introduction

During the last two decades, Machine Learning (ML) has become a key tool in Information Technology and consequently a core of our daily life. At the present time, ML is one of the most used and in-demand technologies of the developing world [1]. According to the «2019 Data Science and Machine Learning Market Study» report

* Corresponding author. Tel.: +34 958-241-000 / Ext. 20466; e-mail: mcarmen@decsai.ugr.es, bacaruiz@ugr.es. At c/ Periodista Daniel Saucedo Aranda, s.n., 18071 Granada, Spain.

[2], data mining [3], advanced algorithms and predictive analytics are ranked one of the most priority projects in business and IT departments.

ML methods have recently contributed to the advancement of forecasting models in order to enhance certain aspects of our daily life [4-6]. As an example, a neural-based classifier can be found in [7] to categorize activities with similar levels and build energy expenditure estimations in order to optimize performance. This study employed radial basis function (RBFN) and generalised regression neural networks (GRNN). But this is merely an instance of many solutions to many other energy-related fields such as ours, electricity demand (ED).

The use of ML models in the electricity realm have shown to highly improve accuracy, robustness and precision with regard to the conventional tools for ED [8] which is key factor for economic and efficient consumption expenditure.

The ED problem lies within the category of time series in the ML scope. In essence, a time-series is a discrete sequence of data points in time [9]. In this study, we focus on the electricity demand, specifically the Spanish territory. These data will be provided to a ML model so as to predict forthcoming consumption using previous values. To do so, we combined ML and time-series techniques. More specifically, we implemented linear regression, regression tree, gradient boosting regression, random forest, multilayer perceptron, long short-term memory neural networks, gated recurrent unit and the Jordan artificial neural network.

The main target of this work is to compare different regression methods to predict electric demand in Spain. Once making these predictions, several metrics will be obtained. Accordingly, we propose the following tasks to attain this goal: 1) data acquisition, 2) cleaning and preprocessing, 3) exploratory analysis, 4) forecasting modelling and 5) comparison and validation of the results.

The ED problem in Spain using time-series has previously been studied by other authors in literature, although their approach has several and significant differences with respect to our proposal. The following lines detail the former studies of ED, with special emphasis on those studies which dealt within the Spanish context.

The first manuscript about this subject can be found in [10]. The authors introduce some scalable methods to predict long-lasting ED time-series in Spain. They suggest Apache Spark framework for distributed computation to accomplish scalability in forecasting methods. Besides, they used MLlib as a machine learning library to implement their models. However, the main MLlib's drawback is that it does not allow multivariable regression which is the main problem of their solution. As a consequence, the authors divided the problem into h sub-problems, where h stands for the number of future steps to compute. They adopted two tree-based techniques of different underlying approach: Gradient-Boosted Trees and Random Forests. In addition, they proposed the use of a Linear Regression method as a reference technique to validate the results. Two metrics were utilized to compare the models: Mean Relative Error (MRE) and Mean Absolute Percentage Error (MAPE).

The second most important study was done by the same authors [11]. This paper introduces a deep learning method to address the problem of big data time-series prognosis [12]. Authors utilized a feed-forward artificial neural network from the H2O framework and the Apache Spark platform to arrange computation in a distributed manner. H2O presents the same problem found in [10], this framework does not allow the implementation of multi-step regression. As a result, the authors elaborated a general-purpose methodology for time-series prediction regardless of the horizon's length. Similarly, the solution consists of solving several forecasting sub-problems

according to how many values are going to be predicted. Therefore, the best model was obtained simultaneously, easing its parallelisation and adaptation to Big Data frameworks.

Some studies endeavour to predict electric load for resources planning as it is a useful task to enhance efficiency in Smart grids. This is straightforward in the case of Hossen et al. [13] who proposed a multi-layered deep neural network to estimate the Spanish electric market. They implemented a TensorFlow model taking into account the MAPE metric for testing weekday and weekend variations. They achieved significant saving with their solution. Blázquez et al.'s research [14] illustrates this point clearly. They analysed the residential demand for electricity in Spain for 47 Spanish provinces and they computed a demand equation for electricity expenditure using a dynamic partial adjustment technique. With their proposal they managed to highlight some of the features of Spanish ED. Similarly, Pérez and Moral [15] presented a method for ED analysis using a simple growth rate decomposition. They recommend their method as a starting point for long-term prognosis, and they use Spain as a case study to compute demand estimations until 2030.

Other techniques have been tested to predict ED in other countries. In a similar case in Australia, Al-Musaylh et al. [16] adopted multivariate adaptive regression spline, SVM and autoregressive integrated moving average (ARIMA) models to predict short-term ED with a 24-h horizon. One can find Grey Models (GM) applied to forecast electricity supply in Turkey [17] where, due to the economic uncertainty, electricity usage shows a chaotic and nonlinear trend. To solve this problem, the authors suggest predicting electricity consumption through the combination of grey predictions and a rolling mechanism, and their solution proved to be more accurate than the model implemented so far. Another interesting research done for Turkey was made by Erdogdu [18] who incorporated ARIMA modelling to co-integration analysis. China [19-23] and Canada [24-26] also present some investigations in this field, such as a neuro-fuzzy approach [25, 27, 28] or other hybrid solutions [20, 21, 23] using neural networks and classic methods.

As discussed above, there are research works in literature where the ED has been studied in many contexts [8, 13-15, 29], e.g., in Spain [10, 11] as in the case of this research. Most of them have developed big data-oriented solutions using classic regression methods or even multi-layer perceptron neural networks.

In contrast to the former studies, this work mainly introduces Recurrent Neural Networks (RNN) [30] in both its original version and its big data-oriented version using Spark, along with classic regression methods in its original version. These methods will be compared with the ones implemented in the aforementioned studies. Hence the final goal of this study is to analyse the most suitable approach for ED in Spain considering several metrics.

The rest of the paper is structured as follows: the proposed methodology is detailed in section 2; section 3 introduces all the experimentation carried out; the next point 4 focuses on describing the results; a discussion about the outcomes obtained is detailed in section 5; and finally section 6 compiles the conclusions achieved from this research.

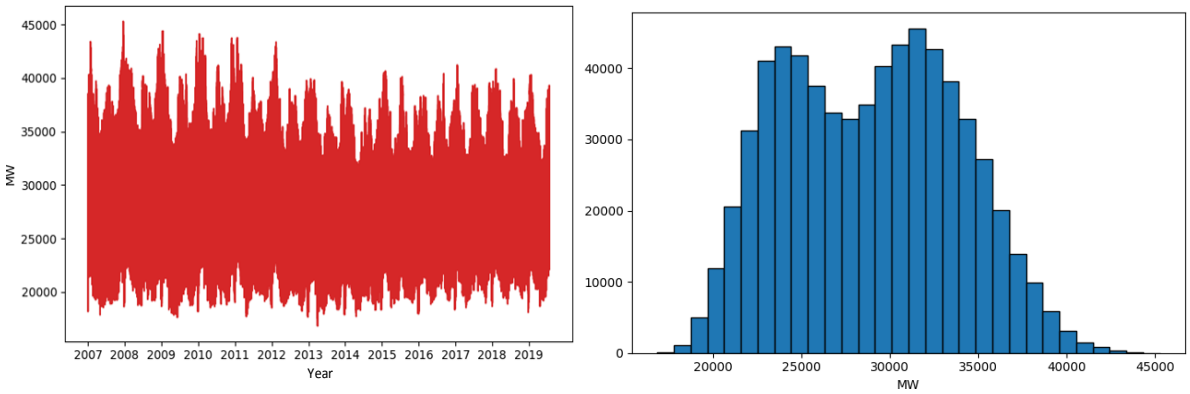
2. Methodology

The current research was developed in three main stages: First, data collection and pre-processing. Second, modelling of the problem and implementation of the proposed models. And finally, validation and analysis of the results.

2.1. Dataset

The dataset used in this work was obtained from the Spanish Electricity Network (SEN) [31]. The official website offers several elements that can be used to extract information from the Spanish ED, such as, a calendar to select a specific day of the ED, a graph for daily demand visualization, a data table to extract the numerical information, accumulated demand from diverse sources of electric energy and the selection of the electric system to be displayed.

In the context of this piece of research, we focus on the first and second tab which provide the data needed for this work. More specifically, we use the following information: date (day and hour), actual (concerning the actual ED at the moment), expected ED (the forecasted ED made by the Spanish Electric Network) and scheduled (this stands for the operational time schedule, i.e., the planned production). All these values are provided in megawatts, and cover a timeline ranging comes from 2007 to 2019. Figure 1a shows all the data over the years and Figure 1b the corresponding histogram which provides information about how common are the values we can find in this ED time-series. From these graphs, we can see how the minimum would be about 15000 and the maximum would be around 45000. Moreover, as can be seen in these figures, the data are arranged close to the mean.

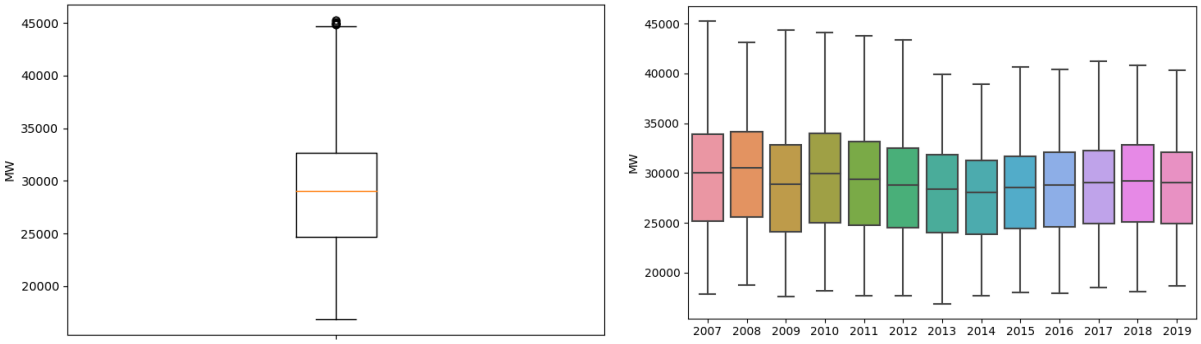


(a) (b)

Figure 1. Illustration of a) the Spanish Electricity Demand and b) its corresponding histogram from 2007 to 2019.

In order to detect outliers, we will display the whole time-series in the same graph and we will do the same for splitting the data into years. This was illustrated in Figure 2a and b respectively. The first plot exhibits some outliers in the top part of the boxplot. These points are fairly close to the maximum which stands for $Q_3 + 1.5 \cdot IQR$ as a consequence we can conclude that they may not be such outliers. And this hypothesis can be supported if we see

the second chart. Figure 2b does not display any outliers. What we can tell from these figures is that the lowest points remain quite stable, and the maximum demand slightly decreases.



(a) (b)

Figure 2. Boxplot of the Electricity Demand for a) the whole period (from 2007 to 2019) and b) each year.

2.2. Preprocessing

After obtaining the Spanish ED data, a preprocessing stage is needed to clean data. The first problem is rather straightforward as there are records that appears more than once, i.e., we have redundant information because of the repeated records. This can be seen from the official website of the Electricity Network at first glance. Another important problem but less obvious is missing values, i.e., there are empty rows, for example.

As a result, our first labour is to solve the problem of duplicate information. To this end, we must consider the following: the data of the ED is in a 10-minutes basis from 21h of the previous day until 03:50 of the next day. This means that there are 18 values of the previous day, 144 values of the present day and 24 values of the next day.

Accordingly, a regular year ought to have 52560 rows, and a leap year should have 52704 values. In our case, we have 3 leap years, 2008, 2012 and 2016. It is important to note that we only extracted data until July 22nd, 2019, which is the 203rd day of the year. Hence, after pre-processing and cleaning the data, one can check the number of rows per year, and we can compare the theoretical values and the actual ones.

Furthermore, some records are empty. To handle this problem, we carried out the called missing-data imputation. First of all, we counted the number of missing items in every column. Only three years (2007, 2008 and 2015) had no data values stored for some of the variables in observation. Since we have three fairly correlated variables, we can utilize this extra information to employ a method such as the k Nearest Neighbours (kNN) to compensate for these values. Provided an instance with at least one of the ED values, one may compute the closest row to complete its empty registers. Finally, once these steps have been completed, we have three univariate time-series according to the «actual», «expected» and «scheduled» electricity demand. In this work context, we mainly focus on the «actual» demand for modelling. Nonetheless, the rest of the columns will be used to compare and validate our results. Additionally, the data were deseasonalised and detrended. We normalised the data and applied a sliding window (see Figure 3) to compute future values, modelling thus a one-step-ahead prediction. After that, the models

were applied to these pre-processed data. Note that errors are calculated after applying all these steps in reverse order, i.e., after de-normalisation and re-trending predictions.

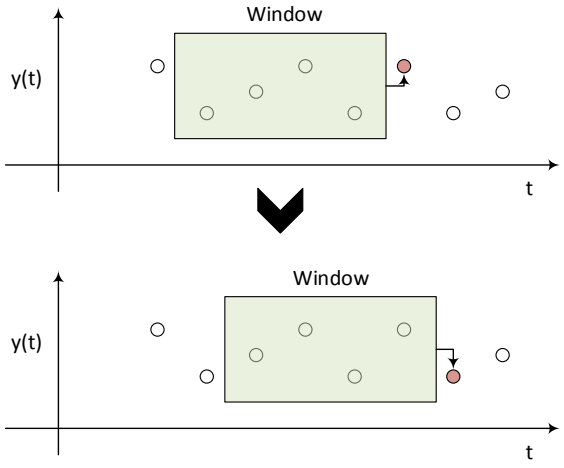


Figure 3. Representation of the sliding window used to preprocess the data.

2.3. Technologies

This section is intended to briefly describe the required methods for the ED forecasting, which include the Linear Regression (LR), Regression Tree (RT), Gradient Boosting Regression (GBR), Random Forest (RF) and Multi-Layer Perceptron (MLP), Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU) and the Jordan Recurrent Neural Network (JRNN).

LR is a statistical method that is widely used for analysis and prediction in the electricity field, amongst others [32]. It has been employed to estimate unknown values using other related variables, also called factors. In this technique, the scalar response is known as a dependent variable, and the rest of them, used for modelling, are identified as independent variables. LR attempts to model relationships between variables by fitting a linear equation to collected data [33].

The second method is one of the most popular ML algorithms due to their simplicity and intelligibility [34]. This predictive model is also used in statistics, along with data mining and ML. It employs a ramified structure to go from observations to conclusions regarding a specific item. The former is commonly called leaves. This method follows a set of splitting rules to divide the predictor variable. It distributes the input into different parts and assigns an estimated value to every region [35].

Thirdly, Gradient Boosting for Regression (GBR) builds an additive predictor in a forward stage-wise manner. In this fashion, GBR optimises the arbitrary differentiable loss function. In each phase, multiclass RTs fit on the negative gradient of the binomial or multinomial deviance loss function [36]. This tree ensemble technique is frequently a set of RT known as CART. It is also called like that as it uses a gradient descent algorithm to minimise loss when adding new RTs [37].

A similar model is Random Forest (RF). This technique is often combined with regression analysis to perform different kinds of prediction, e.g., short-term [34]. It is characterized by feature selection and is an important extension and evolution of bagging algorithms for ensemble learning, i.e., it is a compound of several individual predictors, specifically RT. However, the bottom line of this model is how it splits training data into different samples of the same size after that a particular RT fits them. This sample is called bootstrap and may be chosen several times [38].

Another technique used is the Artificial Neural Network (ANN) [9, 13, 23, 30, 36, 39]. In this study, we implemented four kinds of ANNs. The most common is the Multi-Layer Perceptron (MLP) which is a kind of feedforward ANN. MLP is made of at least three layers of nodes, also called neurons: an input, hidden and output layer. Each node employs a nonlinear activation function but for the input layer, this allows them to model nonlinearity in data.

On the other hand, LSTM has turned out as one of the most popular ANN for sequence data processing. The idea behind LSTM is to memorise the past outputs in memory and use them for successive predictions. This learning of past trends is possible due to some gates together with a memory line added in this model [40]. To do so, LSTM has three gates: input, output and forget.

Prior to explaining the next model, it becomes necessary to briefly introduce the concept of Recurrent Neural Network (RNN). RNNs are a type of artificial neural networks that not only takes the current input values but also it uses previous states of the neuron. Since the recurrent units have information about past values, this makes it possible to provide them with memory, and thus, enabling the model to handle information over time [41].

In contrast, Gated Recurrent Unit (GRU) is a class of ANN with two gates: reset and update. The function of the first ones is very similar to forget gates in LSTM, which were designed to give the memory cells ability to determine when to delete certain knowledge [42]. GRU was proposed as a simplification of LSTM and produce equally excellent results [43]. For that reason, we implemented this model as well.

Lastly, a simpler version of those recurrent neural networks previously mentioned was developed, the Jordan Recurrent Neural Network (JRNN). JRNN extends the MLP with context units in which the output of the network is stored. There is another version where the values of the hidden neurons are saved instead of the output in the context units, in that case, it would be the Elman neural network [9]. These units provide the network with the ability to extract temporal information from historical data in both cases[44]. In this work we will implement the JRNN.

As we mentioned before, we conducted a study using both Big Data and normal implementations of the algorithms in order to compare their performance. In many studies, the Big Data approach has contributed to improving the original algorithm concerning the accuracy in terms of prediction and its scalability [45]. It should be noted that a Big Data solution will provide us a framework to model process big data time series. At this point, a single-core approach is not enough and it becomes necessary to distribute the data and its processing across multiple structures using, for example, a cluster of machines.

3. Experiments

This section attempts to explain the experimentation carried out in this study. It was done using an Intel Core i5-4460 3.20 GHz (4 cores), 8GB RAM, Nvidia GeForce GTX 970, 1TB HDD and 250 GB SSD. Even though the experiments were conducted using a PC, the four cores of the CPU and the available GPU were used to speed up the calculations. It consists of running multiple regression algorithms on the actual ED from 2007 and 2019. The dataset is made up of 660378 samples. The regression algorithms developed belong to standard scientific Python libraries to ease experimental replication, and more specifically Scikit-learn was used to implement classic regression algorithms. The classic regression algorithms adapted to Big Data were implemented with MLlib, Keras was used for neural networks and BigDL which provides ANN for Big Data problems using Spark.

In particular, from Scikit-learn and MLlib we utilized LR, RT, GBR and RF. From Keras, MLP, the CUDA version of LSTM and GRU, and the JRNN. Lastly, we implemented the best models obtained by Keras on BigDL.

We remark that the final goal of this experimentation is to verify what kind of algorithm would get the best results, that is to know if classic regression using Scikit-learn is better than the Big Data-oriented regression algorithms in MLlib, the Keras' ANNs or the same ANNs oriented to Big Data in BigDL.

To train the regression models, the dataset was divided into 70% training and 30% test. Besides, when implementing Keras' models we set 30% of the training data as validation for early stopping. We would like to point out that imputation methods were implemented to remove missing values. These points should not be included in the performance assessment because these methods somehow forecast the missing values. Hence it is expected that the error for such values tends to 0. In our case, we found less than 0,2% missing values (only in 2007, 2008 and 2015), for this reason, it will not influence our result and we can skip this step. Since we will implement the same models for Keras and BigDL, the same number of epochs was defined, i.e., if it takes ten epochs to train a model in Keras, it will take the same iterations in BigDL.

Finally, we have yet to decide what metrics will be used to measure the algorithms' performance. In our case, we adopted four metrics:

Root Mean Square Error (RMSE) is a quadratic scoring rule that measures the average magnitude of the error between prediction and actual value [46] can be calculated as follows:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} y_i^2 - \hat{y}_i} \tag{1}$$

Where y_i are the actual value, $\hat{y}_i$ the estimated value and n the number of samples. The second metric is the Mean Absolute Error (MAE) which measures the average magnitude of the errors in estimations regardless of their direction. This metric means if individuals' differences have the same weight in the average. It is defined in the next equation:

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{2}$$

Both MAE and RMSE provide information about the average forecasting error in models according to the observed variable. They can vary between 0 and ∞ and do not consider the error's direction. The lower value of the metrics, the better performance.

Mean Absolute Percentage Error (MAPE) is a statistical metric that gives information about the accuracy of the forecasting system. It expresses the error size in terms of percentage on actual observations. It is often employed in quantitative prediction methods as it provides a criterion of relative overall fit [47]. Its equation is detailed hereunder:

$$MAPE = 100 \cdot \frac{1}{n} \sum_{i=1}^{n} \frac{|y_i - \hat{y}_i|}{y_i} \tag{3}$$

Finally, we adopted R^2, also called coefficient of determination, for goodness of fit. It is the proportion of variation of the dependent variable explained by explanatory variables. It is commonly used to compute the strength of the relationship in regression [48]. It is defined as:

$$R^2 = \frac{SS_{RES}}{SS_{TOT}} = 1 - \frac{\sum_i^n (y_i - \hat{y}_i)^2}{\sum_i^n (\bar{y}_i - \hat{y}_i)^2} \tag{4}$$

Where $\bar{y}_i$ is the average of the actual values, SS_{RES} is the residual sum of squares and SS_{TOT} the sum of squares of y.

4. Results

In this section, we present the results obtained by each regression algorithm implemented. Due to restrictions of paper length, names in tables were abbreviated and some experiments were summarised. For instance, diverse sizes of the sliding window were tested and we only present the results considering the last 4 hours. Tables provide information about mentioned metrics which are in common, all of them have a subscript tr or ts, and stands for training and test, respectively.

The remaining columns are explained together with the specific algorithm as they are particular for each method. We will not show all the experiments done, but we will focus our attention on the best ones amongst all the experiments. Additionally, in order to put together and compare all the models of the same package, we use P_1, P_2 and P_3 to name first, second or third parameter of the related model, respectively. We remark that not all the models have three parameters, in that case, the associated column would be empty. Table 1 condenses the best results drawn from the whole battery of experiments for every model implemented in Scikit-learn. The first column reveals the experiment number and for the corresponding model. The second column is the window size set for that model. And the two next columns stand for parameters of the different models. As mentioned before, each model has its parameters. In this way, LR was tested changing the number of iterations P_1 and a hyper-parameter to compute the learning rate P_2, keep in mind that our LR was implemented using the SGD optimiser, that is the reason why we adjusted a learning rate in the LR, in particular, this parameter is referring to the initial learning rate for the algorithm. In the case of RT, we tried different max depth of a tree. We varied the number of iterations for the boosting process and the maximum depth of the tree for GBR. Finally, RF was tested modifying the number of trees and its depth.

Likewise, Table 2 gathers the best models using MLlib. LR's parameters were the number of iterations and solver. In the later parameter, 0 stands for the normal equations solver and 1 for the limited-memory BFGS

algorithm. Finally, not to replicate the same information, notice that RT, GBR and RF have the same parameters as in Table 1.

Table 1. Summary of the best results from experiments done using Scikit-learn models (Linear Regression, Regression Tree, Gradient Boosting and Random Forest) along with the time cost of each model.

Exp.	Tam	P_1	P_2	$RMSE_{tr}$	$RMSE_{ts}$	MAE_{tr}	MAE_{ts}	$MAPE_{tr}$	$MAPE_{ts}$	R^2_{tr}	R^2_{ts}	T
LR												
1	288	50	0.015	328.7208	403.4494	227.3449	231.8432	0.8082	0.824	0.9958	0.9922	2
2	288	25	0.015	329.9763	404.5972	229.1941	233.9346	0.8165	0.833	0.9958	0.9922	2
3	288	100	0.015	334.0839	407.1911	232.2885	236.5412	0.8205	0.8361	0.9957	0.9921	2
4	144	50	0.015	355.1405	416.0412	250.8365	244.4454	0.8769	0.8614	0.9951	0.9917	1
5	144	100	0.015	355.675	416.3586	250.9605	244.3789	0.8774	0.8612	0.9951	0.9917	1
RT												
1	144	8	-	312.2878	396.8069	224.8896	222.9741	0.7910	0.7901	0.9962	0.9925	48
2	288	8	-	312.2577	397.5609	224.8609	223.1169	0.7909	0.7907	0.9962	0.9924	99
3	120	8	-	312.8648	398.5422	225.2019	223.2643	0.7921	0.7916	0.9962	0.9924	40
4	72	8	-	313.2506	399.1396	225.6909	223.8552	0.7941	0.7941	0.9962	0.9924	23
5	96	8	-	313.1921	399.3416	225.6597	223.9009	0.7940	0.7942	0.9962	0.9924	31
GBR												
1	288	40	8	253.6752	348.6612	186.8594	188.7435	0.6623	0.6727	0.9975	0.9942	4702
2	144	40	8	258.0805	351.4701	189.8079	191.7291	0.6723	0.6826	0.9974	0.9941	2339
3	120	40	8	262.3728	356.0402	192.7429	194.4626	0.6821	0.6926	0.9974	0.9939	1951
4	96	40	8	264.2656	357.9789	194.2077	196.1796	0.6871	0.6984	0.9973	0.9939	1605
5	72	40	8	265.7438	358.6482	195.4219	197.0673	0.6909	0.7012	0.9973	0.9938	1152
RF												
1	144	100	8	294.6795	376.4441	209.961	205.7714	0.7390	0.7298	0.9967	0.9932	3176
2	288	100	8	294.4902	377.095	209.6186	205.5481	0.7376	0.7287	0.9967	0.9932	6503
3	288	50	8	294.5727	377.132	209.7827	205.7433	0.7383	0.7295	0.9967	0.9932	3254
4	288	75	8	294.649	377.1894	209.8399	205.8385	0.7385	0.7299	0.9967	0.9932	4878
5	144	75	8	295.0017	377.3911	210.1893	206.1736	0.7398	0.7312	0.9967	0.9932	2383

Table 2. Summary of the best results from experiments done using MLlib.

Exp.	Tam	P_1	P_2	$RMSE_{tr}$	$RMSE_{ts}$	MAE_{tr}	MAE_{ts}	$MAPE_{tr}$	$MAPE_{ts}$	R^2_{tr}	R^2_{ts}	T
LR												
1	288	100	0	255.0306	369.4993	174.369	190.4164	0.62	0.6749	0.9975	0.9935	384
2	288	25	0	255.0306	369.4993	174.369	190.4164	0.62	0.6749	0.9975	0.9935	355
3	288	50	0	255.0306	369.4993	174.369	190.4164	0.62	0.6749	0.9975	0.9935	380
4	144	25	0	274.9256	378.6962	189.1822	197.0916	0.669	0.6993	0.9971	0.9931	182
5	144	50	0	274.9256	378.6962	189.1822	197.0916	0.669	0.6993	0.9971	0.9931	205
RT												
1	144	8	-	376.8794	418.1501	268.8565	254.7893	0.937	0.9018	0.9945	0.9916	181
2	288	8	-	378.5353	424.1225	269.9844	255.4362	0.9431	0.9053	0.9945	0.9914	341
3	120	8	-	380.4739	425.1294	271.7144	257.137	0.9477	0.9105	0.9944	0.9913	127
4	96	8	-	380.838	427.4098	272.6688	259.1983	0.9512	0.9184	0.9944	0.9913	147
5	72	8	-	383.8138	427.8019	272.3971	257.6613	0.9489	0.9127	0.9943	0.9912	90
GBR												
1	144	40	8	306.4551	399.4531	229.6932	237.6845	0.8126	0.8443	0.9964	0.9924	237
2	96	40	8	311.13	403.7587	232.6496	239.238	0.8185	0.8476	0.9963	0.9922	359
3	288	20	8	317.7092	407.2906	237.5599	242.4186	0.84	0.8609	0.9961	0.9921	7142
4	120	40	8	310.8609	407.5713	231.8135	239.8807	0.8137	0.8492	0.9963	0.992	707
5	144	20	8	325.6915	407.9497	242.7346	244.9422	0.8577	0.8696	0.9959	0.992	464
RF												
1	288	75	8	349.5005	382.0851	246.9499	229.4487	0.8622	0.8118	0.9953	0.993	815
2	288	100	8	349.3008	382.4324	246.8709	229.3205	0.8622	0.8117	0.9953	0.993	1308

3	288	50	8	349.2775	382.5711	247.3814	229.8821	0.864	0.8135	0.9953	0.993	979
4	144	100	8	343.0349	384.566	247.53	232.0862	0.8686	0.8222	0.9955	0.9929	492
5	288	25	8	351.2803	384.6812	249.1063	231.7701	0.8706	0.8203	0.9953	0.9929	794

In a third battery of experiments, we implemented MLP with several number of hidden layers (P_1) and number of neurons (P_2). In the LSTM, GRU and JRNN, the parameter tested was the corresponding specific units of each model.

Table 3. Summary of the best results obtained from experiments done using Keras' models (Multilayer-Perceptron, Long Short-Term Memory, Gated Recurrent Unit and Jordan Neural Networks).

Exp.	Tam	P_1	P_2	$RMSE_{tr}$	$RMSE_{ts}$	MAE_{tr}	MAE_{ts}	$MAPE_{tr}$	$MAPE_{ts}$	R^2_{tr}	R^2_{ts}	T
MLP												
1	144	2	40	273.4883	366.8139	192.9394	199.8251	0.6776	0.7052	0.9971	0.9936	729
2	144	2	50	277.22	366.9591	195.4519	199.8721	0.6876	0.7094	0.997	0.9936	742
3	144	2	60	271.242	367.0705	189.2467	195.643	0.6669	0.6929	0.9972	0.9935	893
4	144	2	20	277.2662	368.7141	196.6498	202.1508	0.6938	0.716	0.997	0.9935	577
5	96	2	30	276.7211	369.6037	195.4608	200.1907	0.6916	0.7103	0.9971	0.9935	576
LSTM												
1	96	30	-	247.3151	346.8803	171.8198	185.4786	0.6066	0.6542	0.9976	0.9942	2384
2	144	30	-	247.2373	348.2311	172.5325	185.4745	0.6108	0.6561	0.9976	0.9942	2610
3	72	50	-	249.9496	350.3873	174.1121	187.5957	0.6158	0.6644	0.9976	0.9941	1565
4	72	100	-	248.3313	351.7054	172.4375	184.6535	0.6099	0.6534	0.9976	0.9941	2143
5	120	30	-	254.5565	352.8263	175.8142	184.4917	0.6215	0.6529	0.9975	0.994	2183
GRU												
1	288	40	-	244.7777	343.1917	169.8666	180.3092	0.6013	0.6397	0.9977	0.9944	3634
2	144	100	-	240.7929	344.1562	167.7567	180.5029	0.5944	0.6398	0.9978	0.9943	2529
3	288	100	-	240.7945	345.815	167.1048	182.8581	0.5919	0.6479	0.9978	0.9943	3938
4	144	40	-	244.2612	345.8154	169.2886	180.9368	0.5984	0.6397	0.9977	0.9943	2766
5	120	40	-	248.4505	347.7254	172.7622	184.7305	0.6098	0.6542	0.9976	0.9942	2909
JRNN												
1	96	30	-	295.62	394.2566	205.796	206.7689	0.7214	0.7301	0.9966	0.9926	1353
2	120	30	-	295.2136	396.5362	205.5229	207.8183	0.7208	0.7341	0.9966	0.9925	831
3	144	30	-	296.4874	396.8505	206.5011	209.4509	0.7267	0.7413	0.9966	0.9925	989
4	120	40	-	296.4495	397.4892	206.107	209.0169	0.7248	0.7394	0.9966	0.9924	1246
5	72	30	-	298.3369	397.9798	207.6403	209.9763	0.7296	0.7423	0.9966	0.9924	1035

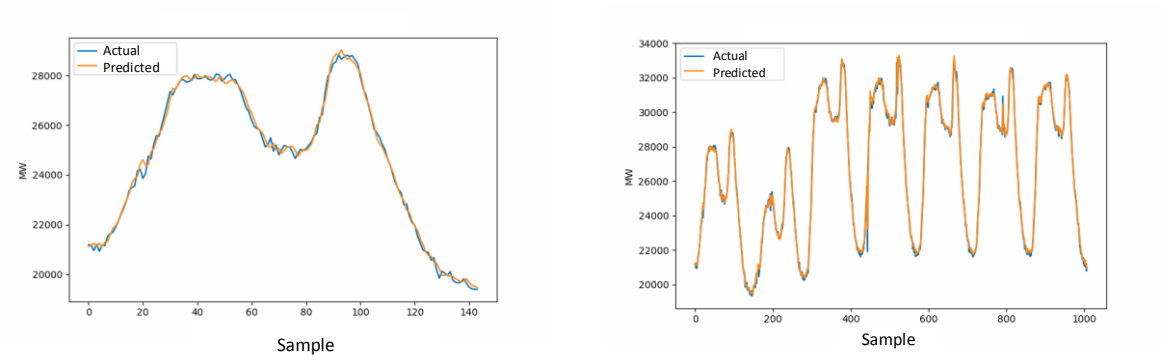
Finally, the last set of experiments was implemented in BigDL using those Keras' models that achieved the best results according to the RMSE in test. As we detailed before, from each metric one can obtain different information. In our case, the benefit of adopting the RMSE as our reference metric is its assessment of the error magnitude and mean error is an indicator of error direction. Besides, RMSE avoids the absolute value and in doing so, one gets the prediction error on the same scale as the output. Furthermore, the other advantage of RMSE is that it punishes large errors, even if the domain is small so that it becomes useful when large errors are particularly undesirable. As can be seen, LSTM and GRU have similar outcomes and are the best ones. Additionally, we selected the best 3 models of each version. These results can be seen in Table 4.

Table 4. Summary of the three best Keras' models implemented in BigDL.

Exp.	Tam	P_1	$RMSE_{tr}$	$RMSE_{ts}$	MAE_{tr}	MAE_{ts}	$MAPE_{tr}$	$MAPE_{ts}$	R^2_{tr}	R^2_{ts}	T
LSTM											
1	72	50	282.2352	367.148	200.7304	201.0771	0.7038	0.708	0.9969	0.9935	6871
2	96	30	277.3078	367.9254	194.1528	197.6941	0.6844	0.7007	0.997	0.9935	7295
3	144	30	315.2163	396.6689	228.2929	229.6589	0.7931	0.804	0.9962	0.9925	9206
GRU											
1	120	40	275.4454	355.2604	191.1378	188.0946	0.6732	0.6668	0.9971	0.994	10217

2	144	40	282.3179	363.0405	198.9298	194.2289	0.7028	0.69	0.9969	0.9937	10579
3	288	40	334.6737	417.159	252.542	240.9043	0.866	0.8504	0.996	0.9916	19263

So far, we have introduced our experiments and results obtained. All these results are discussed in the following section. Additionally, since we have tested many models and representing all of them could be unfeasible, we presented in Figure 4 the best prediction obtained by the GRU neural network using a 288-point window size and 40 units. This figure gathers the results of both daily and weekly predictions.



(a) (b)

Figure 4. Prediction of the best model obtained for a) a representative day and b) its corresponding week.

5. Discussion

We focused our experimentation on implementing a wide range of models to predict the ED in Spain. The first step is to introduce how accurate the SEN forecasts the electricity demand in order to compare and validate our proposal. To do so, we can make use of the 'expected' column provided by the SEN in the website and compute our metrics. These results can be seen in Table 5. Bearing in mind this table, we can now contrast the implemented models of each package and algorithm with the SEN's. In this work, we set an importance order to evaluate how good a model is. Firstly, we focus on the RMSE. Secondly, the MAPE metric. Next, MAE. And finally, R^2. In all cases, we will analyse those metrics acquired from test data. In doing so, to determine what model is better, we will use $RMSE_{ts}$. If two models draw with this metric, we will use MAPE and so on.

Table 5. Statistics obtained from the Spanish Electricity Network's predictions.

RMSE	MAPE	MAE	R^2
471.6463	0.8802	249.8638	0.9909

Table 1, Table 2, Table 3 and Table 4 show the results with the best model per algorithm and package respectively. In these tables the best models are already sorted best to worst. As can be seen from these four tables,

all the models make better predictions than the SEN. All the models get, for every metric in test, better results than the one provided by the SEN.

Another interesting aspect is that all the window sizes tested appear in these tables but for the smallest values, i.e., 24 and 48. From this, one can ascertain that the smaller windows size, the worse results provide.

Once we compared our best models with the ones deployed by the SEN, the next step is to analyse the performance of our models. If we compare all the rows corresponding to a particular model, in all the metrics, it is very difficult to see a great difference between the best result and the other four-best results, i.e., they are fairly similar. Likewise, if we compare all the models we can see a small difference between the best performance attained by the Keras' GRU algorithm with a RMSE of 343,1917 and the last one in the ranking which would be MLlib's RT with a 418,1501. The later has the highest difference of RMSE at only 21.84%. In contrast, SEN's error increases by 37.43% comparing with our best models, which is virtually twice the difference between the best and SEN, and our worst and SEN. Hence, in our worst-best case, there would be an increasing difference in error of a 12.79%.

At first glance, one may conclude that the recurrent neural networks and MLP are the best models. Indeed, from these tables, we can anticipate that Keras' GRU and LSTM were the best ones, which are recurrent networks. Therefore, the first hypothesis was right. Nonetheless, JRNN is another recurrent model as well, but it did not obtain as good results as its counterparts. This result may be a consequence of two factors. The first one, because of the fast convergence of the network; as can be seen in the previous tables, it takes shorter time to train the JRNN than the other neural networks. And secondly, which is linked with the previous reason, it may be owing to its structure, i.e., the JRNN's architecture is also called «simple» recurrent neural network as it uses only one value of the input and output in the recurrent units. This method got the firth-worst position after Scikit-learn's RT and LR, and MLlib's RT and GBR. On the contrary, further analysis of the results showed that MLP is the firth-best model which is an acceptable position.

On the other hand, BigDL's GRU and LSTM reported good predictions too. However, they did not reach the accuracy of Keras' solutions. Both Keras and BigDL utilized exactly the same architecture, hyper-parameters and epochs. In spite of that, BigDL attained slightly worse predictions in comparison with Keras.

The most surprising aspect of these results is in the Scikit-learn's GBR which achieved the third-best error, overcoming those estimations from BigDL and MLP.

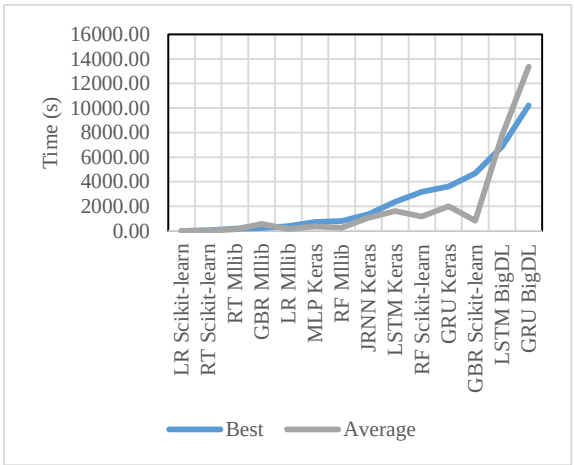
Finally, as a representative example of how good our best model predicted, Figure 4 showed the prediction of a day (Figure 4a) and a week (Figure 4b). At first glance, one could say that both series are virtually the same and we achieved excellent performance. Even though this claim seems to be true, we would like to objectively analyse these two graphs. The first aspect to pay attention is the high range of the data. Since the Spanish ED in this day and week ranges from 20000MW to 34000MW, a small variation apparently does not change the results, and yet it could turn out an unexpectedly sharp increase in energy. Having said that, we can conclude according to the aforementioned metrics that we managed to model the Spanish ED with very good accuracy, and it can be seen in these figures.

5.1. Time cost analysis

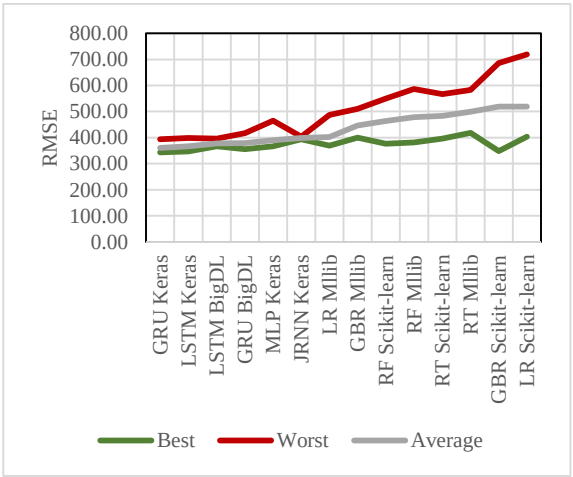
The previous section is devoted to evaluating the models according to their accuracy and error. Getting the best accuracy is usually the most important goal when choosing a model. Nonetheless, this point loses interest if the selected algorithm takes too long to provide results. Thus, in some events, a faster method is a better option and this is why we implemented our models using a Big Data framework. To do so, we analyse the time cost which is the column T showed in the previous tables. Table 6 shows in a ranking-oriented view how long the different algorithm take. The first column is the name of the algorithm, the second one stands for the training time obtained by the most accurate model, and the last column corresponds to the average time spent to train all the models.

Table 6. Running time in seconds of the most accurate model (second column) and the average time (third column) per model and framework.

Algorithm	Time (s)		RMSE		
	Best	Average	Best	Worst	Average
LR Scikit-learn	2,93	0,89	403,45	719,35	519,40
RT Scikit-learn	48,64	27,86	396,81	566,67	482,95
RT Mllib	181,55	128,00	418,15	583,38	499,85
GBR Mllib	237,20	562,93	399,45	510,26	446,16
LR Mllib	384,86	162,79	369,50	486,91	402,50
MLP Keras	729,99	365,68	366,81	465,03	390,03
RF Mllib	815,87	264,84	382,09	586,83	478,05
JRNN Keras	1353,94	1067,83	394,26	403,25	398,58
LSTM Keras	2384,69	1618,62	346,88	398,48	366,92
RF Scikit-learn	3176,08	1174,21	376,44	549,81	463,62
GRU Keras	3634,52	1996,45	**343,19**	**393,70**	**360,91**
GBR Scikit-learn	4702,07	839,18	348,66	685,87	518,44
LSTM BigDL	6871,72	7790,67	367,15	396,67	377,25
GRU BigDL	10217,09	13353,00	355,26	417,16	378,49



(a)



(b)

Figure 5. Summary of the a) running time in seconds and b) the corresponding RMSE for each model.

What is striking in Table 6 and Figure 5 is the rapid response of the Scikit-learn's LR. On average, it takes less than a minute for a 660378-instances training set. This is as a consequence of the SGD optimiser which is extremely fast in problems with more than 10000 samples. Nevertheless, this algorithm provides one of the worst results, being the second last, although its predictions remain relatively good. On the other hand, those algorithms that take longer are the BigDL ones. On average, LSTM took approximately 2 hours and GRU 4 hours. By contrast, these two techniques ranked the third and fourth position in terms of error as can be seen in Figure 5b.

6. Conclusions

This work proposed the implementation of different regression models to predict the Spanish electricity demand. To deal with this problem, a wide variety of models and experiments have been developed. The present results are significant in all the cases as they achieved better prediction than the one estimated by the Spanish Electricity Network. In the worst case our models attained an error 12% better than the SEN, up to a 37% in the best case. The results showed that Keras' neural networks were the best models, in particular those with GRU and LSTM units. By contrast, LR turned out to be the one with highest error but it was the fastest algorithm using the SGD optimizer; it took 0,89 seconds on average. However, LR in MLlib needed more time than the Scikit-learn version, although its accuracy was better. In terms of ANN, MLP was the fastest neural network and obtained an acceptable error. On the other hand, ensemble algorithms (GBR and RF) in Scikit-learn reported to be more accurate than MLlib's version.

Finally, research questions that could be asked include hyper-parameters tuning and multi-step predictions using recurrent neural networks, that is to say, multiple estimations at once using the same window. Regarding hyper-parameters, we conducted our experiments using the classical trial and error to find the best ones, however, we can find other modern approaches [49] in the literature that we recommend adopting in future works like the optimisation algorithm inspired by the COVID-19 [50]. Additionally, implementing more complex architectures may be an interesting improvement in terms of accuracy.

7. Acknowledgments

This work has been developed with the support of the Department of Computer Science and Artificial Intelligence of the University of Granada, TIC111.

8. Abbreviations

ANN	Artificial Neural Network

ARIMA	Autoregressive Integrated Moving Average
DM	Data Mining
ED	Electricity Demand
GBR	Gradient Boosting Regression
GM	Grey Model
GRU	Gated Recurrent Unit
JRNN	Jordan Recurrent Neural Network
kNN	k Nearest Neighbours
LoR	Logistic Regression
LR	Linear Regression
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MAPE	Mean Absolute Percentage Error
ML	Machine Learning
MLP	Multi-Layer Perceptron
MRE	Mean Relative Error
RF	Random Forest
RMSE	Root Mean Square Error
RNN	Recurrent Neural Networks
RT	Regression Tree
SEN	Spanish Electricity Network
SVM	Support Vector Machine

9. Bibliography

1. Rahul, K., et al., *Machine Learning Algorithms for Big Data Analytics*, in *Computational Methods and Data Engineering*. 2020, Springer. p. 359-367. [http://link.springer.com/book/10.1007%2F978-981-15-6876-3#about]
2. Services, D.A., *Data Science and Machine Learning Market Study.* Wisdom of Crowds' Series, 2019. [http://gumroad.com/l/dTfno]
3. Martínez-Álvarez, F., et al., *A Survey on Data Mining Techniques Applied to Electricity-Related Time Series Forecasting.* Energies, 2015. **8**(11). [http://dx.doi.org/10.3390/en81112361]
4. Li, J.H., et al., *Effects of light-emitting diodes under capped daily energy consumption with combinations of electric power and photoperiod on cultivation of Chlorella pyrenoidosa.* Bioresource Technology, 2016. **205**: p. 126-132. [http://dx.doi.org/10.1016/j.biortech.2016.01.041]
5. Rueda, R., et al., *An Ant Colony Optimization approach for symbolic regression using Straight Line Programs. Application to energy consumption modelling.* International Journal of Approximate Reasoning, 2020. **121**: p. 23-38. [http://dx.doi.org/10.1016/j.ijar.2020.03.005]
6. Alvarez, F.M., et al., *Energy Time Series Forecasting Based on Pattern Sequence Similarity.* IEEE Transactions on Knowledge and Data Engineering, 2011. **23**(8): p. 1230-1243. [http://dx.doi.org/10.1109/TKDE.2010.227]

7. Lin, C., et al., *A Wearable Sensor Module With a Neural-Network-Based Activity Classification Algorithm for Daily Energy Expenditure Estimation.* IEEE Transactions on Information Technology in Biomedicine, 2012. **16**(5): p. 991-998. [http://dx.doi.org/10.1109/TITB.2012.2206602]
8. Eseye, A.T., et al., *Machine Learning Based Integrated Feature Selection Approach for Improved Electricity Demand Forecasting in Decentralized Energy Systems.* IEEE Access, 2019. **7**: p. 91463-91475. [http://dx.doi.org/10.1109/ACCESS.2019.2924685]
9. Ruiz, L.G.B., et al., *Energy consumption forecasting based on Elman neural networks with evolutive optimization.* Expert Systems with Applications, 2018. **92**(Supplement C): p. 380-389. [http://dx.doi.org/10.1016/j.eswa.2017.09.059]
10. Galicia, A., et al. *Scalable Forecasting Techniques Applied to Big Electricity Time Series*. 2017. Cham: Springer International Publishing. [http://dx.doi.org/10.1007/978-3-319-59147-6_15]
11. Torres, J.F., et al., *A scalable approach based on deep learning for big data time series forecasting.* Integrated Computer-Aided Engineering, 2018. **25**: p. 335-348. [http://dx.doi.org/10.3233/ICA-180580]
12. Galicia, A., et al., *Multi-step forecasting for big data time series based on ensemble learning.* Knowledge-Based Systems, 2019. **163**: p. 830-841. [http://dx.doi.org/https://doi.org/10.1016/j.knosys.2018.10.009]
13. Hossen, T., et al. *Short-term load forecasting using deep neural networks (DNN)*. in *2017 North American Power Symposium (NAPS)*. 2017. [http://dx.doi.org/10.1109/NAPS.2017.8107271]
14. Blázquez, L., N. Boogen, and M. Filippini, *Residential electricity demand in Spain: New empirical evidence using aggregate data.* Energy Economics, 2013. **36**: p. 648-657. [http://dx.doi.org/10.1016/j.eneco.2012.11.010]
15. Pérez-García, J. and J. Moral-Carcedo, *Analysis and long term forecasting of electricity demand trough a decomposition model: A case study for Spain.* Energy, 2016. **97**: p. 127-143. [http://dx.doi.org/10.1016/j.energy.2015.11.055]
16. Al-Musaylh, M.S., et al., *Short-term electricity demand forecasting with MARS, SVR and ARIMA models using aggregated demand data in Queensland, Australia.* Advanced Engineering Informatics, 2018. **35**: p. 1-16. [http://dx.doi.org/10.1016/j.aei.2017.11.002]
17. Akay, D. and M. Atak, *Grey prediction with rolling mechanism for electricity demand forecasting of Turkey.* Energy, 2007. **32**(9): p. 1670-1675. [http://dx.doi.org/10.1016/j.energy.2006.11.014]
18. Erdogdu, E., *Electricity demand analysis using cointegration and ARIMA modelling: A case study of Turkey.* Energy Policy, 2007. **35**(2): p. 1129-1146. [http://dx.doi.org/10.1016/j.enpol.2006.02.013]
19. Wang, Y., et al., *Application of residual modification approach in seasonal ARIMA for electricity demand forecasting: A case study of China.* Energy Policy, 2012. **48**: p. 284-294. [http://dx.doi.org/10.1016/j.enpol.2012.05.026]
20. Zhu, S., et al., *A seasonal hybrid procedure for electricity demand forecasting in China.* Applied Energy, 2011. **88**(11): p. 3807-3815. [http://dx.doi.org/10.1016/j.apenergy.2011.05.005]
21. Bao, G., et al., *Hybrid Short-term Load Forecasting Using Principal Component Analysis and MEA-Elman Network*, in *Intelligent Computing Methodologies: 12th International Conference, ICIC 2016, Lanzhou, China, August 2-5, 2016, Proceedings, Part III*, D.-S. Huang, K. Han, and A. Hussain, Editors. 2016, Springer International Publishing: Cham. p. 671-683. [http://dx.doi.org/10.1007/978-3-319-42297-8_62]
22. Xie, Y. and Q. Weng, *Detecting urban-scale dynamics of electricity consumption at Chinese cities using time-series DMSP-OLS (Defense Meteorological Satellite Program-Operational Linescan System) nighttime light imageries.* Energy, 2016. **100**: p. 177-189. [http://dx.doi.org/10.1016/j.energy.2016.01.058]
23. Qin, S., et al., *A hybrid model based on smooth transition periodic autoregressive and Elman artificial neural network for wind speed forecasting of the Hebei region in China.* International Journal of Green Energy, 2016. **13**(6): p. 595-607. [http://dx.doi.org/10.1080/15435075.2014.961462]
24. Douthitt, R.A., *An economic analysis of the demand for residential space heating fuel in Canada.* Energy, 1989. **14**(4): p. 187-197. [http://dx.doi.org/10.1016/0360-5442(89)90062-5]

25. Zahedi, G., et al., *Electricity demand estimation using an adaptive neuro-fuzzy network: A case study from the Ontario province – Canada.* Energy, 2013. **49**: p. 323-328. [http://dx.doi.org/10.1016/j.energy.2012.10.019]
26. Farhat, A.A.M. and V.I. Ugursal, *Greenhouse gas emission intensity factors for marginal electricity generation in Canada.* International Journal of Energy Research, 2010. **34**(15): p. 1309-1327. [http://dx.doi.org/10.1002/er.1676]
27. Yang, X., F. Yu, and W. Pedrycz, *Long-term forecasting of time series based on linear fuzzy information granules and fuzzy inference system.* International Journal of Approximate Reasoning, 2017. **81**: p. 1-27. [http://dx.doi.org/10.1016/j.ijar.2016.10.010]
28. Sadaei, H.J., et al., *Short-term load forecasting method based on fuzzy time series, seasonality and long memory process.* International Journal of Approximate Reasoning, 2017. **83**: p. 196-217. [http://dx.doi.org/10.1016/j.ijar.2017.01.006]
29. Wijaya, T.K., T. Ganu, and D. Chakraborty, *Consumer segmentation and knowledge extraction from smart meter and survey data.* Proceedings of the 2014 International Conference on Data Mining, 2014: p. 226-234. [http://dx.doi.org/10.1137/1.9781611973440.26]
30. Ruiz, L.G.B., M.I. Capel, and M.C. Pegalajar, *Parallel memetic algorithm for training recurrent neural networks for the energy efficiency problem.* Applied Soft Computing, 2019. **76**: p. 356-368. [http://dx.doi.org/10.1016/j.asoc.2018.12.028]
31. Network, S.E., *Spanish Electricity Demand in real-time.* 2020. [http://demanda.ree.es/visiona/home]
32. Saber, A.Y. and A.K.M.R. Alam. *Short term load forecasting using multiple linear regression for big data*. in *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*. 2017. [http://dx.doi.org/10.1109/SSCI.2017.8285261]
33. El Mouatasim, A., *Simple and Multi Linear Regression Model of Verbs in Quran.* American Journal of Computational Mathematics, 2018. **8**(1): p. 68-77. [http://dx.doi.org/10.4236/ajcm.2018.81006]
34. Liu, J. and Y. Li, *Study on environment-concerned short-term load forecasting model for wind power based on feature extraction and tree regression.* Journal of Cleaner Production, 2020. **264**: p. 121505. [http://dx.doi.org/10.1016/j.jclepro.2020.121505]
35. Upadhya, S.S. and A.N. Cheeran, *Performance Comparison of Regression Techniques In Predicting Parkinson Disease Severity Score Using Speech Features.* Biomedical Engineering: Applications, Basis and Communications, 2018. **30**(04): p. 1850025. [http://dx.doi.org/10.4015/s1016237218500254]
36. Sboev, A., et al., *Evaluation of the Cardiovascular Risk in Middle-aged Workers: An Artificial Neural Networks-based Approach.* Procedia Computer Science, 2016. **80**: p. 2418-2422. [http://dx.doi.org/10.1016/j.procs.2016.05.540]
37. Punmiya, R. and S. Choe, *Energy Theft Detection Using Gradient Boosting Theft Detector With Feature Engineering-Based Preprocessing.* IEEE Transactions on Smart Grid, 2019. **10**(2): p. 2326-2329. [http://dx.doi.org/10.1109/TSG.2019.2892595]
38. Lahouar, A. and J. Ben Hadj Slama, *Day-ahead load forecast using random forest and expert input selection.* Energy Conversion and Management, 2015. **103**: p. 1040-1051. [http://dx.doi.org/10.1016/j.enconman.2015.07.041]
39. Umematsu, T., et al. *Improving Students' Daily Life Stress Forecasting using LSTM Neural Networks*. in *2019 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI)*. 2019. [http://dx.doi.org/10.1109/BHI.2019.8834624]
40. Siami-Namini, S. and A.S. Namin, *Forecasting economics and financial time series: ARIMA vs. LSTM.* arXiv preprint, 2018. [http://arxiv.org/abs/1803.06386]
41. Davis, A., et al., *Neural attention mechanisms for malware analysis*. 2017, Google Patents. [http://patents.google.com/patent/US9705904B1/en]
42. Lee, K., et al. *CNN and GRU combination scheme for Bearing Anomaly Detection in Rotating Machinery Health Monitoring*. in *2018 1st IEEE International Conference on Knowledge Innovation and Invention (ICKII)*. 2018. [http://dx.doi.org/10.1109/ICKII.2018.8569155]

43. Goodfellow, I., et al., *Deep learning*. Vol. 1. 2016: MIT press Cambridge. [http://synapse.koreamed.org/upload/SynapseData/PDFData/1088HIR/hir-22-351.pdf]
44. Malleswaran, M., V. Vaidehi, and N. Sivasankari, *A novel approach to the integration of GPS and INS using recurrent neural networks with evolutionary optimization techniques.* Aerospace Science and Technology, 2014. **32**(1): p. 169-179. [http://dx.doi.org/10.1016/j.ast.2013.09.011]
45. Pérez-Chacón, R., et al., *Big data time series forecasting based on pattern sequence similarity and its application to the electricity demand.* Information Sciences, 2020. **540**: p. 160-174. [http://dx.doi.org/https://doi.org/10.1016/j.ins.2020.06.014]
46. Zheng, B., et al. *A Hybrid Machine Learning Model for Range Estimation of Electric Vehicles*. in *2016 IEEE Global Communications Conference (GLOBECOM)*. 2016. [http://dx.doi.org/10.1109/GLOCOM.2016.7841506]
47. Sutheebanjard, P. and W. Premchaiswadi. *Stock Exchange of Thailand Index Prediction Using Back Propagation Neural Networks*. in *2010 Second International Conference on Computer and Network Technology*. 2010. [http://dx.doi.org/10.1109/ICCNT.2010.21]
48. Kasuya, E., *On the use of r and r squared in correlation and regression.* Ecological Research, 2019. **34**(1): p. 235-236. [http://dx.doi.org/10.1111/1440-1703.1011]
49. Torres, J.F., et al., *Deep Learning for Time Series Forecasting: A Survey.* Big Data, 2020. [http://dx.doi.org/10.1089/big.2020.0159]
50. Martínez-Álvarez, F., et al., *Coronavirus Optimization Algorithm: A Bioinspired Metaheuristic Based on the COVID-19 Propagation Model.* Big Data, 2020. **8**(4): p. 308-322. [http://dx.doi.org/10.1089/big.2020.0051]