Süleyman Demirel Üniversitesi
Fen Bilimleri Enstitüsü Dergisi
Cilt 23, Sayı 2, 635-646, 2019

Süleyman Demirel University
Journal of Natural and Applied Sciences
Volume 23, Issue 2, 635-646, 2019

DOI: 10.19113/sdufenbed.494396

A Comparative Study of Machine Learning and Deep Learning for Time Series Forecasting: A Case Study of Choosing the Best Prediction Model for Turkey Electricity Production

Ramazan ÜNLÜ *[1]

Gümüşhane Üniversitesi, İktisadi ve İdari Bilimler Fakültesi, Yönetim Bilişim Sistemleri Bölümü, 29000, Gümüşhane
(ORCID: https://orcid.org/0000-0002-1201-195X)

(Alınış / Received: 10.12.2018, Kabul / Accepted: 30.07.2019, Online Yayınlanma / Published Online: 30.08.2019)

Keywords
Machine learning,
Deep learning,
LSTM,
Time series

Abstract: Over the last decades, Turkey pays special attention to electricity production to afford its needs. Researchers applied different methodologies including statistical-based and artificial intelligence-based to correctly predict the future amount of electricity production, consumption, and demand. However,limited researchers focused on Turkey's electricity production prediction problem as a time series analysis. For this reason, we tackle this problem by considering it as a time series analysis in this study. We have used different methods including traditional machine learning algorithms Support Vector Regression (SVR) and Multilayer Perceptrons (MLP) and a deep learning algorithm Long Short-Term Memory (LSTM) to create a better model for Turkey monthly electricity production dataset. Based on our findings LSTM outperforms SVR and MLP approaches in terms of commonly used statistical error evaluation metrics

Zaman Serileri Tahminlenmesinde Makine Öğrenimi ve Derin Öğrenme Tekniklerinin Kıyaslanması: Türkiye Elektirik Üretimi için En İyi Tahmin Modelinin Seçilmesine Yönelik Bir Vaka Çalışması

Anahtar Kelimeler
Makine öğrenimi,
Derin öğrenme,
UKDH,
Zaman serileri

Özet: Son yıllarda Türkiye ihtiyaçlarını karşılayabilmek adına elektrik üretimine yoğun bir şekilde dikkat vermektedir. Araştırmacılar elektrik üretim, tüketim ve talep mikarını doğru bir şekilde tahmin etmek için istatistik ve yapay zeka tabanlı yöntemleride içeren birçok farklı metod uygulamışlardır. Sınırlı sayıda araştırmacı Türkiye'nin elektrik üretim tahminleme problemini bir zaman serisi analizi olarak irdelemiştir. Bu nedenle bu çalışmada söz konusu problem zaman serileri analizi olarak ele alınmıştır. Bu açıdan çalışmada hem Destek Vektör Makineleri (DVM) ve Çok Katmanlı Nöronlar (ÇKN) gibi klasik makine öğrenimi yöntemleri hem de Uzun Kısa Dönemli Hafıza (UKDH) yöntemi gibi derin öğrenme yöntemi Türkiye'nin üretmesi gereken aylık elektrik üretim miktarını tahmin etmek için kullanılmıştır. Çalışmanın bulgularına dayalı olarak derin öğrenme algoritması istatistiksel hata oranlarına göre diğer klasik makine öğrenimi yöntemlerinden daha başarılı sonuçlar vermektedir.

1. Introduction

Prediction of energy sources is an important issue for the governments, energy sector investors and other related corporations. Creating a sufficient prediction model for energy production, consumption or demand can help future planning regarding energy sustainability which is a global issue. Many countries have various sources to produce energy to be able to afford their mandatory needs. While developed countries have much more alternatives such as natural gas, nuclear, wind, solar energy sources, etc. , developing countries, for example, Turkey, have limited sources to produce the required amount of energy. That's why future planning in terms of the amount of energy is becoming more vital to allocate potential sources in an optimum way. Thus, researchers have been focused on generating a better prediction model for energy context over the decades.

When we talk about the energy, the first thing comes to mind is possibly the electricity which might be produced from many different sources and is a fundamental power source for almost everything used in our daily life. Therefore, electricity production planning is crucial for many countries as well as Turkey. Turkey's electricity production and demand planning studies go back to 1960s led by State Planning Organization. Before sophisticated prediction methods proposed or well improved, simple statistical methods like regression method are used for future planning over the years.

* *Corresponding author: ramazanunlu@gumushane.edu.tr*

Not only public or official enterprises focus on to solve the aforementioned problem, but studies in the literature show researchers used the various methods to create electricity prediction model from electricity data. However, researchers more interested in specifically electricity consumption and demand in Turkey because of government and their related branches put major emphasis on modeling and predicting electricity consumption. From this context, we can find various studies in the literature taking care of solving the problem. In the study of Dilaver and Hunt, a good grouping of method description developed to create a prediction model is given for electricity data [1]. Those are categorized as 1) causality studies, 2) relationship studies, and 3) forecast studies. Because our motivation is creating a better prediction model we give our attention to the forecast studies. Also, as we mentioned above, there can be found many different studies around the world giving prominence to the forecasting model by using various electricity dataset belongs to different regions or countries. And yet our study specific to Turkey, we focus more on researches used historical Turkey Electricity datasets.

In the study of Unler, Particle Swarm Optimization and Ant Colony algorithm are used with different features such as population, import, and export amount [2]. Ant Colony Optimization for the purpose of generating and demanding the amount of electricity is also used by [3]. Another prediction technique called Grey Theory (GT) [4] which can solve uncertain systems and imperfect information to forecast Turkey's electricity demand [5]. Similarly, Hamzacebi and Es used optimized GT to predict Turkey's annual electricity consumption [6].

Autoregressive Integrated Moving Average (ARIMA) and seasonal ARIMA are used in the study of [7]. They have used the methods not only to create a baseline prediction model but also use them for future forecasting. Besides this study, there exist various researches using statistical methods such as ARIMA and Seasonal ARIMA [8–10]. In addition to those specific papers, more detailed information can be found in the survey paper proposed by [11].

As well as classical statistical methods, researchers have also used more sophisticated techniques such as Machine Learning Algorithms. Support Vector Machine (SVM) is used by [12] in which the electricity consumption of Turkey is modeled as a function of four socioeconomic indicators, namely population, GNP, imports and exports. Artificial Neural Network (ANN) is another advance method utilized for energy-related datasets [13, 14]. In those studies using the ANN method, a shallow network is created for low dimensional datasets (i.e. ANN with one hidden layer fed by 3 dimensional 32 samples [13]). As in the study of [12], some others utilized different attributes forming raw data and generate to find energy demand or production instead of using the known energy values itself as time series [14]. Another approach preferred is Fuzzy logic employed by [15]. Instead of using multiple attributes, gross domestic product (GDP) and energy demand were selected as input and output parameters respectively.

On the other hand, some researches convert electricity data itself to the times series data to create a prediction model without needing any additional information [1, 16, 17]. In the context of artificial intelligence, studies analyzing electricity data as a time series are limited compared to the classical methods. Time series analysis with a univariate dataset (i.e. having only electricity production data) is not as easy as using a tabular dataset. Based on our literature review, articles proposed between 2005-2018 focuses on Turkish Electricity production, demand, and consumption forecasting problem, but there exist limited studies approaching the problem from a time-series perspective. On the other hand, studies which take the problem as a time series analysis have too bounded details on the data preprocessing step [1, 16, 17]. More details and foundations of the studies about Turkey's electricity prediction task can be found in the study of [1]. Also, notwithstanding advanced methods such as SVR and ANN are well-performed techniques for the prediction problems, more sophisticated methods such as deep learning methodologies are not applied so far. That's why it can be thought of as another gap in the literature related to the used methods.

Machine learning (ML) models have been widely used and new algorithms are developed in order to handle with unsupervised and supervised learning problems over the last decades [18–20]. During the recent years, deep learning models have taken a lot of attention and various deepest structures such as a deep neural network (DNN), deep belief network(DBN), recurrent neural network(RNN), and so on are proposed and make a huge impact on the field. Although the core objective of those methods is the same which is revealing hidden patterns from much more complex dataset, the systematic of algorithms and targeted data structure might be different. Generally speaking, a DNN is similar to the MLP except that it might have many hidden layers. Thanks to the advance in computational technology, it is possible to solve a neural network system consisting of many hidden layers. Due to its deepest structure scholars called it as deep neural network. It is also the general name of deeper neural network structures [21]. On the other hand, some deep neural network approaches such as DBN is designed for a more specific purpose. DBF is a generative model and it is trained by a series of stacked Restricted Boltzmann Machines (RBMs). While a regular DNN is a feed-forward neural network DBN has the undirected connection between some layers. And, those undirected layers are trained using an unsupervised learning algorithm [22]. The reason behind training some layers is to obtain good generalization while the number of hidden layers increase in which regular DNN overfits [23]. Another deep learning approach is RNN which is designed to be trained on sequenced datasets. As different from DNN and DBN, it has recurrent connections between units [24]. As we mentioned above, RNN is specifically designed for the sequential datasets and it achieves a state of art performance on important problems such as natural language processing [25], speech recognition [26], and machine translation [27].

For aforementioned reasons, we have used three different methodologies including machine learning algorithms SVR and ANN and Deep learning method LSTM with the purpose of comparison of chosen methods based on error rates for Turkey electricity production dataset.

The remainder of this article is organized as follows. In Section 2, the details of the dataset and data preparation step are described. In Section 3, we provide the results of chosen methods in terms of given error evaluation metrics. In final Section 4 we conclude and discuss our study as well as a projection for future studies.

2. Material and Method

As different from statistical methods in a time series analysis, we need to prepare the dataset to effectively use a machine learning algorithm. To do this, we can unroll the process in three steps 1) restructure dataset 2) build the model of experiment 3) run the model. The followed workflow is shown in Figure 1. Therefore, we have organized the section in a way that explains each step in details. In the next Section 2.1, we give the details and the visualization of the dataset. Through the experiment, all data samples are normalized. On the other hand, all the methods are compared based on evaluation metrics are given in Table 1. These are commonly used evaluation metrics used for the regression problems.

Table 1. Evaluation metrics and corresponding formulations.

The formulation of evaluation metrics
Mean Absolute Error (MAE) $\frac{1}{n}\sum_{i=1}^{n}\lvert y-\tilde{y}\rvert$
Root Absolute Error (MSE) $\frac{\sum_{i=1}^{n}(\bar{y}-y)^2}{n}$
Root Relative Squared Error (RMSE) $\sqrt{\frac{\sum_{i=1}^{n}(\bar{y}-y)^2}{n}}$
Correlation Coefficient (R^2) $\left(\frac{n\sum(y\tilde{y})-\sum y\sum\tilde{y}}{\sqrt{n(\sum y^2)-(\sum y)^2}-\sqrt{n\sum(\tilde{y}^2)-(\sum\tilde{y})^2}}\right)^2$

2.1. Preperation of the dataset

In our experiment, we have used a real data set obtained from TEIAS (Turkish Electricity Transmission Corporation) website https://www.teias.gov.tr to apply and compare machine learning algorithms. The data set shows monthly electricity production from January 1975 to December 2017. It is a univariate sequential data as shown in Figure 2.

We can see in Figure 2 the data is not stationary. In other words, there exist an increasing trend in the data and that does not make predictions easy. Therefore, we should remove trends from the data. One easiest way of doing that is differencing the data. That can verbally be described as the observation from the previous time step (t-1) is subtracted from the current observation (t). Finally, we will have differenced data. The following Figure 3 shows the data without any trend.

Removing trends from the data make it easy to analyze. However, before utilizing chosen algorithms we need to prepare the dataset to which chosen algorithms can be applied.

To simply illustrate the process of the preparation step, assume that we are given a dataset $X = \{x_i\}_n^m$, where $x_i \in R$, which is suitable to utilize a machine learning algorithm. This dataset can be illustrated as in matrix Equation 1. In this dataset, input $x_{11}, x_{12}, \ldots, x_{1m}$ yields the output y_1, input $x_{21}, x_{22}, \ldots, x_{2m}$ yields the output y_2 and so on. However, a time series dataset does not look like the described one above. Therefore, we need to reconstruct it to apply a machine learning algorithm. A toy time-series data or sequential data $X = \{x_i\}$, where $x_i \in R$, which is also a column vector shown in the matrix below.

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} & \ldots & x_{1m} \\ x_{21} & x_{22} & x_{23} & \ldots & x_{2m} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & x_{n3} & \ldots & x_{nm} \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \tag{1}$$

As different from the previous dataset, the sequence of the observations is crucial and highly influence the result of the forecast problem. We can think x_2 is the result of x_1, x_3 is the results of x_2 and so on.

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \tag{2}$$

To reconstruct the dataset, we might create some lags which can be thought as the input data. There is no clear answer to the question of how many lags need to be created to get the optimum result. However, the grid search method can be applied to find the best possible solution. The dataset above can be reconstructed based on creating 1 lag as shown in the following matrix.

$$\begin{bmatrix} x_1 & x_2 \\ x_2 & x_3 \\ \vdots & \vdots \\ x_n & NaN \end{bmatrix} \tag{3}$$

In the example above, the raw dataset is the input data and created lag has become the output data. One needs to note that the last data sample x_n does not yield an output because the result of it is not given. In that point, x_n can be used as input to forecast x_{n+1} and x_{n+1} is used to forecast x_{n+2} and so on. This given toy example is for the univariate time series and we can extend this procedure for the multivariate dataset. So, let say we are given raw data consist of two features as shown in the matrix below.

$$\begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ \vdots & \vdots \\ x_{n1} & x_{n2} \end{bmatrix} \tag{4}$$

As following the procedure above, the reconstructed dataset will be as shown in the matrix below. Features

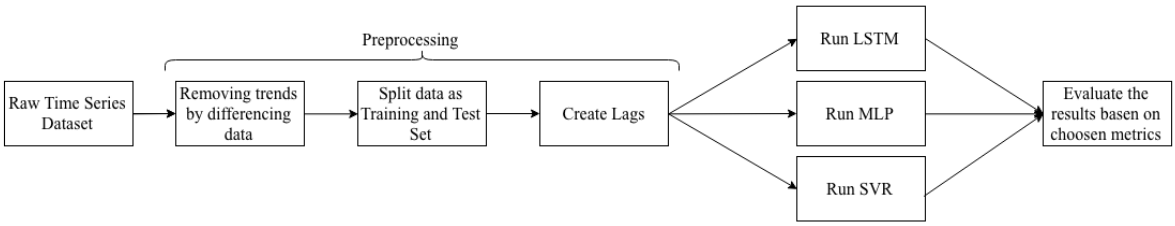

Figure 1. Illustration of the work-flow.

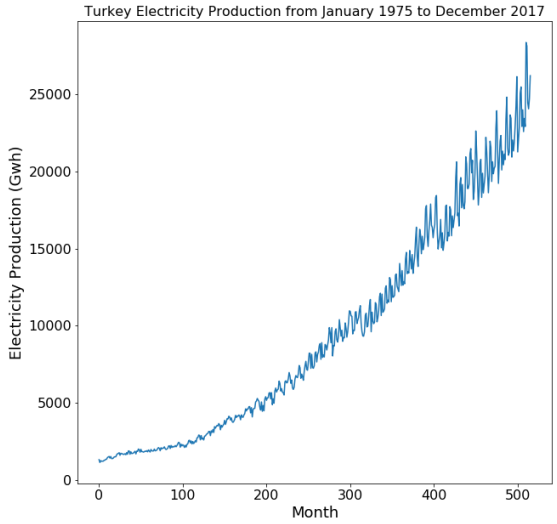

Figure 2. Turkey electricity production from January 1975 to December 2017. There are 516 months between these dates.

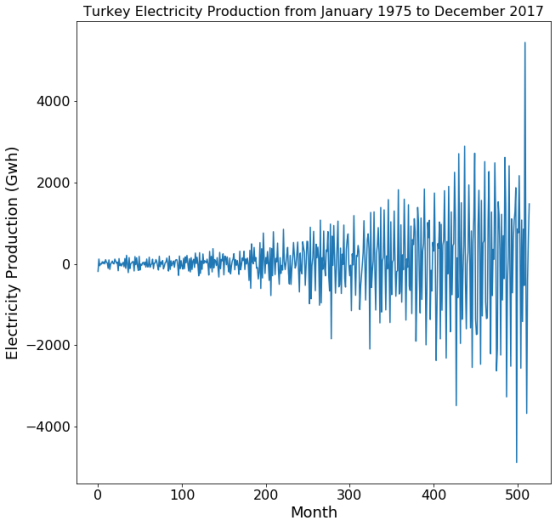

Figure 3. The dataset transformed by differencing process.

in the raw dataset desired to be forecasted will be the output of reconstructed dataset, and all other features will be the input data.

$$\begin{bmatrix} x_{11} & x_{12} & x_{21} & x_{22} \\ x_{21} & x_{22} & x_{31} & x_{32} \\ \vdots & \vdots & \vdots & \vdots \\ x_{(n-1)1} & x_{(n-1)2} & x_{n1} & x_{n2} \\ x_{n1} & x_{n2} & NaN & NaN \end{bmatrix} \tag{5}$$

One of the necessities of the construction process to be able to run the experiment is removing NaN values which are created as a result of unknown prior values. After removing them we are ready to apply any machine learning algorithm and as we mentioned above we utilized one classical ML method and one Deep Learning methods given in Section 2 in details.

After following the described procedure, we can now transform the dataset given at the beginning of the section. To concretely illustrate, assume we have created 1 lag and 4 lags for the given dataset. Following matrices show the reconstructed electricity dataset used through our experiment. We represent the only head of the dataset for readability convenience.

$$\begin{bmatrix} -181.29 \\ 122.40 \\ -42.5 \\ 7.0 \\ -1.60 \\ 54.90 \\ 11.79 \\ 55.10 \\ 4.5 \\ 106.59 \\ \vdots \end{bmatrix} \xrightarrow{\text{1-lag}} \begin{bmatrix} -181.29 & 122.40 \\ 122.40 & -42.5 \\ -42.5 & 7.0 \\ 7.0 & -1.60 \\ -1.60 & 54.90 \\ 54.90 & 11.79 \\ 11.79 & 55.10 \\ 55.10 & 4.5 \\ 4.5 & 106.59 \\ 106.59 & 42.1 \\ \vdots & \vdots \end{bmatrix}$$

$$\begin{bmatrix} -181.29 \\ 122.40 \\ -42.5 \\ 7.0 \\ -1.60 \\ 54.90 \\ 11.79 \\ 55.10 \\ 4.5 \\ 106.59 \\ \vdots \end{bmatrix} \xrightarrow{\text{4-lags}} \begin{bmatrix} -181.29 & 122.40 & -42.5 & 7.0 \\ 122.40 & -42.5 & 7.0 & -1.60 \\ -42.5 & 7.0 & -1.60 & 54.90 \\ 7.0 & -1.60 & 54.90 & 11.79 \\ -1.60 & 54.90 & 11.79 & 55.10 \\ 54.90 & 11.79 & 55.10 & 4.5 \\ 11.79 & 55.10 & 4.5 & 106.59 \\ 55.10 & 4.5 & 106.59 & 42.1 \\ \vdots & \vdots & & \end{bmatrix}$$

The last column of the constructed data is the output of the corresponding sample. For instance, in the case of creating 4 lags, the output of the sample $[-181.5, 122.40, -42.5]$ is $[7.0]$, the output of sample $[122.40, -42.5, 7.0]$ is $[1.60]$ and so on. By doing this, our dataset is constructed to be suitable for the application of any machine learning method.

2.2. Algorithms

Through this study, we are being motivated to create an algorithmic pool to make a better comparison. For this reason, we chose three different algorithms Support Vector Regression (SVR), Multilayer Perceptrons (MLP), Long Short Term Memory (LSTM) with the aim of having a different mathematical theory behind them. By doing this, we can bring individual strengths of the algorithms into the forefront. Each of them is explained in details during the following subtitles.

Support Vector Regression: Support vector machines(SVMs) is a well-known supervised learning algorithm proposed by [28]. The method is first design for the binary classification problem which can be linearly separable. Then it is extended to use for the regression problems named as Support Vector Regression (SVR) [29]. To mathematically formulate it assume we have a training dataset$(x_1, y_2), \ldots, (x_l, y_l)$, where each $x_i \in R^n, y_i \in R$ the decision function is given by Equation 6.

$$f(x) = w\phi(x) + b \tag{6}$$

With respect to $w \in R^n$ and $b \in R$, where ϕ denotes a nonlinear mapping from R^n to high dimensional space.
To ensure $f(x)$ is as flat as possible, we need to find it with the minimal norm value as shown in Equation 7.

$$J(w) = \frac{1}{2}||w|| \tag{7}$$

Subject to all residuals having a value less than ε; or, in equation form:

$$w\phi(x_i) + b - y_i \leq \varepsilon \tag{8}$$

We can infer that it is not possible to meet this condition for the all points. So, we can add slack varibles ξ^+ and ξ^- to provide some flexibility and rewrite the formulations as shown below in Equation 9:

$$J(w) = \frac{1}{2}||w|| + C\sum_{i}^{n}(\xi^+ + \xi^-) \tag{9}$$

subject to:

$$\begin{aligned} y_i - (w\phi(x_i) + b) &\leq \varepsilon + \xi^+ \\ (w\phi(x_i) + b) - y_i &\leq \varepsilon + \xi^- \\ \xi^+ &\geq 0 \\ \xi^- &\geq 0 \end{aligned}$$

where C is a constant value that control the penalty values imposed to the variable which lies outside the ε margin and help to avoid being overfitting. Finally, we can compute the loss function that ignores the error if the predicted value is less than or equal to ε. Thus, it can be formulated as below

$$L_\varepsilon = \begin{cases} 0, & \text{if } |(w\phi(x_i) + b) - y_i| \leq \varepsilon \\ |(w\phi(x_i) + b) - y_i| - \varepsilon, & \text{otherwise} \end{cases} \tag{10}$$

For the mathematical convenience, the optimization problem described above can be solved in dual form.

Multi Layer Perceptrons: Artificial neural network or called Multilayer perceptrons (MLP) is commonly used method to retrive the nonlinear relation from the data [30–33]. It is structed by stacked fully connected neurons or called perceptrons. It consists some layers named as input,hidden, and output as shown in Figure 4. Data enters to the system from input layers, and input values are forwarded to neurons in the next layer. The value of the each neuron is the linear combination of the values of the nodes from the previous layer. This process is called as feed forwerded neural networks. Then the system works from output layer to the input layer to omptimize weights by taking partial derivatives. This method is named as back propagation. To mathematically describe, the output of n^{th} neuron in l^{th} layer is calculated as the linear combination of the previous layer such that:

$$o = y_l^n = w^T x + b \tag{11}$$

where w^T is the connection weight The value of the node is transformed by an activation function. There exist different activation functions in the literature. We have used the sigmoid one which transfom the value as being 1 or 0 based on the Equation 12.

$$\sigma(x) = \frac{1}{1 - e^{-(w^T x + b)}} \tag{12}$$

After calculation the value of each node, we need to adjust the weights that minimize the error which can be calculated based on the Equation 13 for given dataset $D = \{(x_1, t_1), (x_2, t_2), \ldots, (x_d, t_d), .., (x_m, t_m)\}$

$$E[\vec{w}] = \frac{1}{2}\sum_{d \in D}(t_d - o_d) \tag{13}$$

Finally to adjust w_i as $w_i := w_i + \Delta w_i$, we need to utilize following partial derivatived procedure simultaneously for each wi.

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i} \tag{14}$$

where η is the learning rate, Δw_i is the adjusment value. After taking the derivatives, we can wrap up the adjustment rules as shown in Equation 15

$$\Delta w_i = -\eta \sum_{d \in D}(t_d - o_d)x_{id} \tag{15}$$

One needs to note that, because our problem is a regression problem, there will be only one neuron in the output layer of the MLP model.

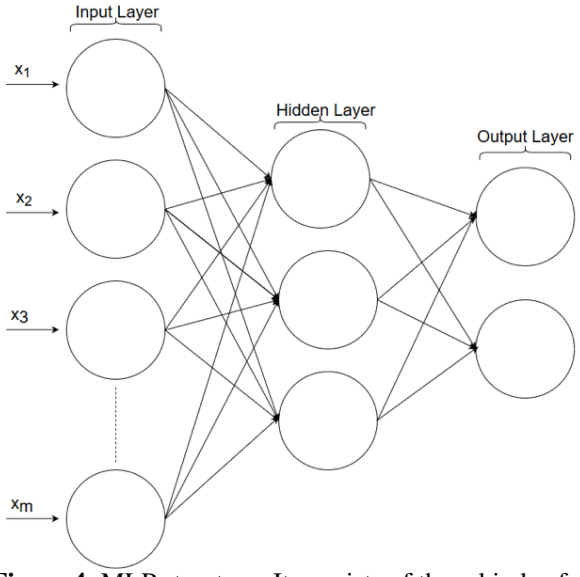

Figure 4. MLP structure. It consists of three kinds of layers; Input, Hidden, and Output layers.

Long-Short Term Memory: Long Short-Term Memory (LSTM) networks are an extension for recurrent neural networks (RNN) developed by [34]. The neural network described in the previous title has forward connections. Each layer connects to the next layer and final hidden layer connects to the output layer. As different from the traditional neural network, LSTM is well suited to remember what is important things learned from experiences previously occurred. To start from very beginning, assume have a given sequential dataset $X = \{x^{(i)<t>}, y^{(i)<t>}\}$ where i represent i^{th} example, t is the position of the sample in a sequence and $y^{(i)<t>}$ is the true output for the i^{th} example in t position. To understand LSTM structure in details , we first need to look at the RNN and Gated Recurrent Unit (GRN). Assume we are given a dataset $X = \{x^{<1>}, x^{<2>}, \ldots, x^{<t>}, \ldots, x^{<T_x>}\}$ (i.e X is a sentence and $x^{<t>}$ is the t^{th} word in that sentence). RNN takes the information from $x^{<t>}$ and activation value $a^{<t-1>}$ from the previous time step to help prediction with $y^{<t>}$. The simple structure of an RNN is shown in Figure 5.

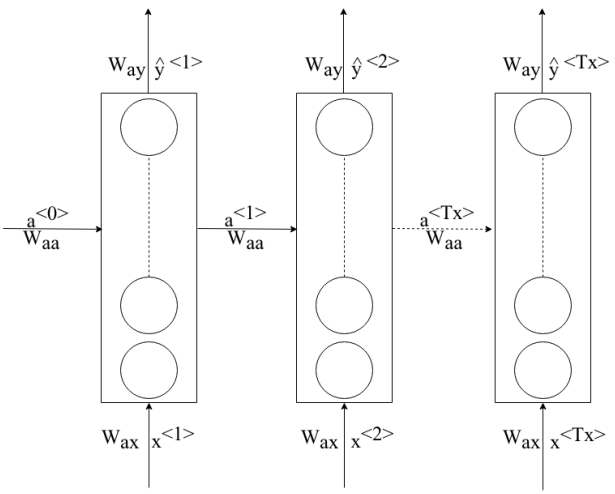

Figure 5. Simple RNN structure for sequential dataset.

Based on the structure of the RNN we can build the formulation of the RNN as shown in Equations 16 and 17.

$$a^{<t>} = g(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a) \tag{16}$$

where $a^{<t>}$ is the activation value in time step t, g is the choosen activation function, W_{aa} is the parameter for the activation values, W_{ax} is parameters for the input values, an b_a is the bias value. By using the value of $a^{<t>}$, the prediction of the corresponding sample $y^{<t>}$ can be made based on Equation 17.

$$\hat{y}^{<t>} = g(W_{ya}a^{<t>} + b_y) \tag{17}$$

where $W_{ya}a$ is the paremeters for the activations and b_y is the bias value.
After forward feeding the neural network, the parameters can be optimized using backpropagation algorithm with the aim of minimizing loss function which is given in Equations 18 and 19.

$$L^{<t>}(\hat{y}^{<t>}, h^{<t>}) = -y^{<t>}log\hat{y}^{<t>} - (1-y^{<t>})log\hat{y}^{<t>} \tag{18}$$

This is the Loss value for the specified sample $x^{<t>}$. Thus, we can calculate overall loss value based on following Equation 19

$$L(\hat{y}^{<t>}, h^{<t>}) = \sum_{t=1}^{T_x} (\hat{y}^{<t>}, y^{<t>}) \tag{19}$$

Now, we can go through the GRU before explaining the LSTM structure. Basic GRU cell structure is given in Figure 6

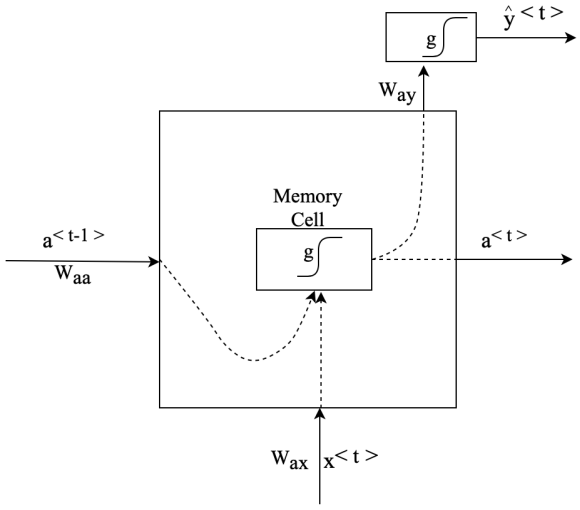

Figure 6. Simple RNN structure for sequential dataset.

GRU is the modification of the RNN hidden unit. The idea behind the GRU is remembering valuable information belongs to the input data until using that. So, assume that $c^{<t>}$ which is equal to $a^{<t>}$ is the memory cell in the neural network structure used to save some crucial information. We need to decide over every time step whether to change the value of $c^{<t>}$. In other words, we need to ask the question of whether we should update the information reserved still need to be remembered. To do this, we first

need to calculate candidate memory cell value $\hat{c}^{<t>}$ as shown in Equation 20.

$$\hat{c}^{<c>} = g(W_{cc}c^{<t-1>} + W_{cx}x^{<t>} + b_c) \tag{20}$$

where g is the activation function, W_{cc} and W_{cx} the parameters for the memory cell values and input data respectively, $c^{<t-1>}$ is the memory cell value from previous time step. The crucial problem in this point deciding if that is the correct time to update the memory cell value. To decide this, we define a parameter Γ_u named as update gate which takes either 1 or 0 meaning "update" and "do not update" respectively. The formulation of Γ_u is as in Equation 21.

$$\Gamma_u = g(W_{ux}c^{<t-1>} + W_{ux}x^{<t>} + b_u) \tag{21}$$

where g is the activation function, W_{uc} and W_{ux} are the paremeters for memory cell value and input data respectively during the update calculation. Finally $c^{<t>}$ value can be overwrited as in Equation 22

$$c^{<t>} := \Gamma_u\hat{c}^{<t>} + (1-\Gamma_u)c^{<t-1>} \tag{22}$$

Note that if the Γ_u is equal to 1 $c^{<t>} = \hat{c}^{<t>}$ meaning that update $c^{<t>}$, otherwise $c^{<t>} = c^{<t>}$ meaning that do not update. The whole process of GRU is illustrated as in Figure 6.

We can now go forward to LSTM which is an even slightly more powerful and more general version of GRU. The assumption $c^{<t>} = a^{<t>}$ is no longer valid anymore. Also, we define two new parameters Γ_f and Γ_o named as forget gate and output gate respectively. Revised formulation of the LSTM is shown in Equations 23,24,25,26,27 and 28.

$$\hat{c}^{<t>} = g(W_{ca}a^{<t-1>} + W_{cx}x^{<t>} + b_c) \tag{23}$$

$$\Gamma_u = g(W_{ua}a^{<t-1>} + W_{ux}x^{<t>} + b_u) \tag{24}$$

$$\Gamma_f = g(W_{fa}a^{<t-1>} + W_{fx}x^{<t>} + b_f) \tag{25}$$

$$\Gamma_o = g(W_{oa}a^{<t-1>} + W_{ox}x^{<t>} + b_o) \tag{26}$$

$$c^{<t>} = \Gamma_u\hat{c}^{<t>} + \Gamma_f c^{<t-1>} \tag{27}$$

$$a^{<t>} = \Gamma_o c^{<t>} \tag{28}$$

As we can see that rule of overwriting $c^{<t>}$ based on Equation 22 is not valid anymore, instead we imply the formula given in Equation 23 and activation $a^{<t>}$ is calculated based on Equation 28 instead of equaling it to $c^{<t>}$. The simple structure of LSTM cell is shown in Figure 7.

2.3. K-Fold Cross-Validation

K-fold cross-validation is a method which optimizes the calculation time and variance. The input data as randomly are parted to k groups (named as fold). Each group is used for test and remaining for the training processes [35]. In other words, the algorithm tries to learn and test itself in k times. Finally, the performance of the method is calculated

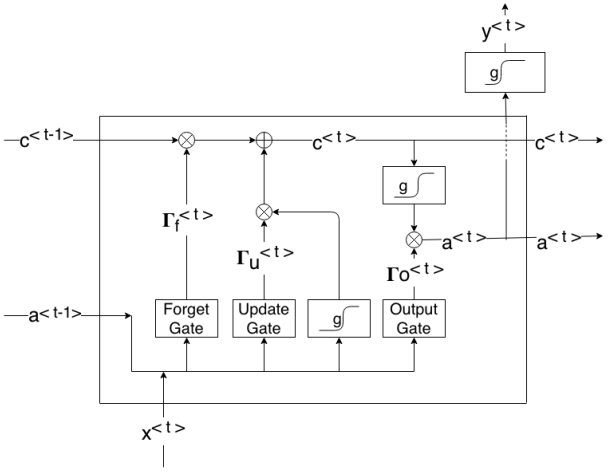

Figure 7. The LSTM cell structure for the sequential dataset. $\bigotimes$ and $\bigoplus$ represents element-wise multiplication and summation processes respectively.

as the average evaluation metric (i.e. average accuracy for the classification problem) [36]. Cross-validation structure is illustrated in Figure 8. Now, assume we are given a dataset D which is randomly split k different subsets $(D_1, D_2, \ldots D_k)$ such that:

$$\bigcup_{i=1}^{k} D_1, D_2, \ldots, D_k = D \tag{29}$$

During the prediction process, as we mentioned above each D_i is saved for the testing and remaining is saved for the training process. It is repeated until each subset of the data is used as a test set (i.e. for the 10-folds cross-validation this process repeated 10 times). Finally, the performance of the predictor is calculated as the average of k runs.

$$p* = \frac{\sum_{i=1}^{k} p_i}{k} \tag{30}$$

where $(p_i, i = 1, 2, \ldots k)$ is the performance of predictor in i^{th} iteration.

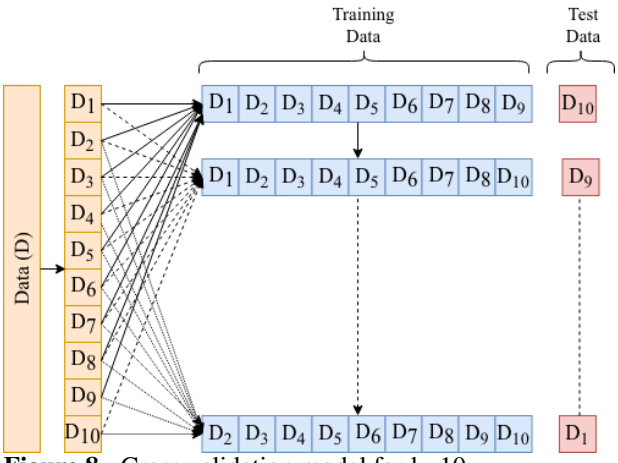

Figure 8. :Cross validation model for k=10.

3. Results

In this section, we provide results of all chosen algorithms and comparison of them in terms of selected error metrics.

As we mentioned in Section 2, there is no clear way to decide how many lags to get the best possible solution. Therefore, we have created lags ranging from 1 to 20 for comparison purpose. All experiments are conducted with Python version of 3.6 with Scikit-learn and Keras libraries and performed on Intel Core i7, 2.7 GHz with 16 of RAM in a 64-bit platform. Finally, the dataset is normalized as usual before running the algorithms.

Figure 9 shows the comparison of all three methods concerning MAE error metrics. As it can be seen that the LSTM method gives a better result than the other two methods MLP and SVR. The SVR and LSTM method behave similarly in terms of the number of lags. With the small number of lags, they do not learn well enough, and error decreases while the number of lags increases to some point. After that, the SVR method does not enhance itself and LSTM performance starts to decrease.

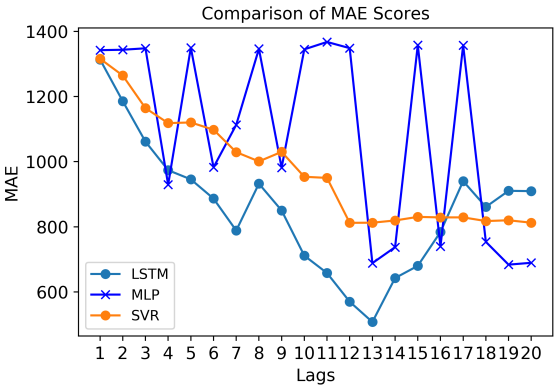

Figure 9. Comparison of MAE scores.

As it should be expected the same thing is happening for other error metrics, the following Figures 10 and 11 shows the performance of methods in terms of MSE and RMSE.

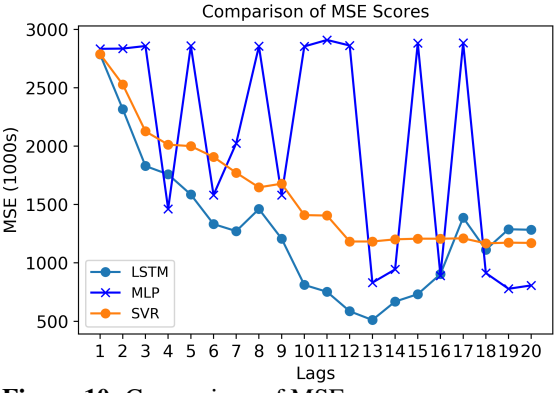

Figure 10. Comparison of MSE scores.

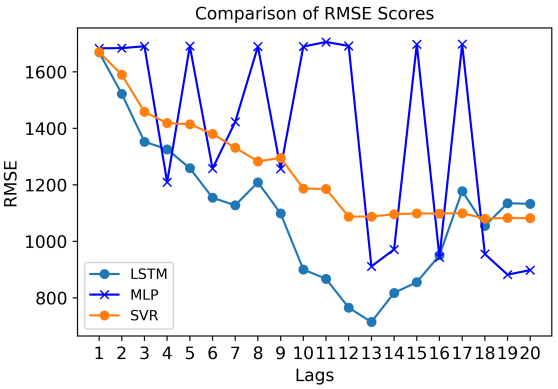

Figure 11. Comparison of RMSE scores.

We can see the same pattern through the number of lags. LSTM method again outperforms the other two methods. On the other hand, MLP gives the best result using some number of lags such as lag=4, however, there is no robustness through the different lags. It is highly sensitive to the number of lags so that might be a crucial problem in terms of choosing the optimum lag number. On the contrary, LSTM and SVR are more robust. They learn better and better by increasing the number of lags to some point. After that, the SVR method stays stable and LSTM starts to yields poor result.

The reason behind that might be an effect of the small and high number of lags in the learning process. With the small number of lags, the performance of each algorithm is very poor because algorithms do not learn enough from back in the history of the data. On the other hand, with the high number of lags provide to algorithms chance of learning from the past of the data enough. However, going back too much will misguide methods due to the meaningful relationship between the current and past data will be diminished. Thus, choosing the number of lags such that giving enough information from the past while keeping its influence to current time maximum helps to get the possible result.

By using maximum meaningful information back in the history of the data can yield the predictions which mostly related to current time data called ground true outputs. The following Figure 12 shows the R^2 score between the predictions of all three methods and known true values. LSTM method captures the best relation in general in comparison to the other two methods.

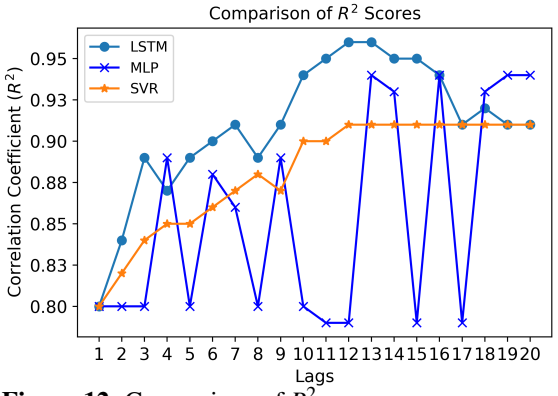

Figure 12. Comparison of R^2 scores.

So far, we give the comparison figures to create a visually better understanding. The following Tables 2 , 3, and 4 show overall results of methods MLP, SVR, and LSTM respectively in terms of MAE, MSE, RMSE , and R^2 evaluation metrics in details for each lags. The best performance for each metric is bolded. As we pointed out before, the number of lags highly effect the performance of the methods. While MLP does not produce a stable results as the number of lags increase, SVR and LSTM leverage their performance until some points. Then, their performance reduce because of high dimension of the data (i.e increasing number of lags increases the dimension of data).

Table 2. The error rates of MLP method for different lags.

Lags	MAE	MSE	RMSE	R^2
Lag-1	1342.16	2833446.8	1683.28	0.8
Lag-2	1343.47	2835498	1683.89	0.8
Lag-3	1347.89	2857622.99	1690.45	0.8
Lag-4	929.04	1462929.51	1209.52	0.89
Lag-5	1349.86	2859035.66	1690.87	0.8
Lag-6	982.47	1581702.33	1257.66	0.88
Lag-7	1112.97	2025346.21	1423.15	0.86
Lag-8	1346.33	2855400.56	1689.79	0.8
Lag-9	981.49	1580526.44	1257.19	0.89
Lag-10	1344.66	2853948.82	1689.36	0.8
Lag-11	1367.36	2909293.26	1705.67	0.79
Lag-12	1348.41	2861808.26	1691.69	0.79
Lag-13	688.34	831057.75	911.62	0.94
Lag-14	737.31	944260.15	971.73	0.93
Lag-15	1357.55	2882366.3	1697.75	0.79
Lag-16	739.27	891317.4	944.1	0.94
Lag-17	1357.32	2884360.06	1698.34	0.79
Lag-18	753.69	912913.83	955.47	0.93
Lag-19	**683.69**	**778465.28**	**882.31**	**0.94**
Lag-20	689.38	807424.36	898.57	**0.94**

Table 3. The error rates of SVR method for different lags.

Lags	MAE	MSE	RMSE	R^2
Lag-1	1315.89	2788474.6	1669.87	0.8
Lag-2	1264.34	2527743.54	1589.89	0.82
Lag-3	1164.25	2126850.81	1458.37	0.84
Lag-4	1118.25	2012922.23	1418.77	0.85
Lag-5	1120.23	2000213.65	1414.29	0.85
Lag-6	1097.92	1907059.41	1380.96	0.86
Lag-7	1029.1	1772126.83	1331.21	0.87
Lag-8	1000.68	1646982.14	1283.35	0.88
Lag-9	1029.92	1678488.57	1295.56	0.87
Lag-10	953.42	1409385.62	1187.18	0.9
Lag-11	950.04	1405610.85	1185.58	0.9
Lag-12	812.08	1183322.2	1087.81	0.91
Lag-13	812.48	1183361.07	1087.82	0.91
Lag-14	819.33	1201854.56	1096.29	0.91
Lag-15	830.28	1207597.39	1098.91	0.91
Lag-16	828.73	1207207.95	1098.73	0.91
Lag-17	828.82	1210439.38	1100.2	0.91
Lag-18	817.5	1167423.28	1080.47	0.91
Lag-19	819.88	1173491.32	1083.28	0.91
Lag-20	812.53	1171159.29	1082.2	0.91

Table 4. The error rates of LSTM method for different lags.

Lags	MAE	MSE	RMSE	R^2
Lag-1	1313.49	2786239.3	1669.2	0.8
Lag-2	1185.48	2316925.14	1522.14	0.84
Lag-3	1061.45	1829924.74	1352.75	0.89
Lag-4	973.74	1758285.52	1326	0.87
Lag-5	945.5	1585439.01	1259.14	0.89
Lag-6	886.48	1332626.53	1154.39	0.9
Lag-7	788.37	1272171.72	1127.91	0.91
Lag-8	932.25	1461408.05	1208.89	0.89
Lag-9	850.01	1207662.77	1098.94	0.91
Lag-10	711.33	811535.25	900.85	0.94
Lag-11	657.95	752604.17	867.53	0.95
Lag-12	569.92	586025	765.52	0.96
Lag-13	507.6	511054.27	714.88	0.96
Lag-14	718.88	825223.67	817.74	0.94
Lag-15	679.99	732359.11	855.78	0.95
Lag-16	783.9	904106.27	950.85	0.94
Lag-17	940.17	1387581.99	1177.96	0.91
Lag-18	860.96	1113159.15	1055.06	0.92
Lag-19	910.37	1288234.26	1135	0.91
Lag-20	909.39	1283639.83	1132.98	0.91

In addition to given error figures and tables, we provide the following Figures 13, 14, and 15 illustrating the actual and predicted values for each method. Due to the high number of lags which ranging from 1 to 20, we only illustrate the one that gives the minimum RMSE score.

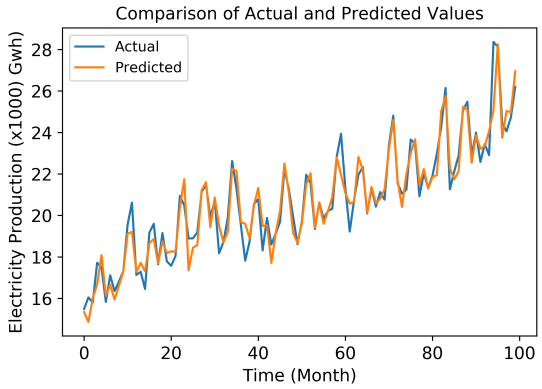

Figure 13. Comparison of actual values and predicted values by LSTM .

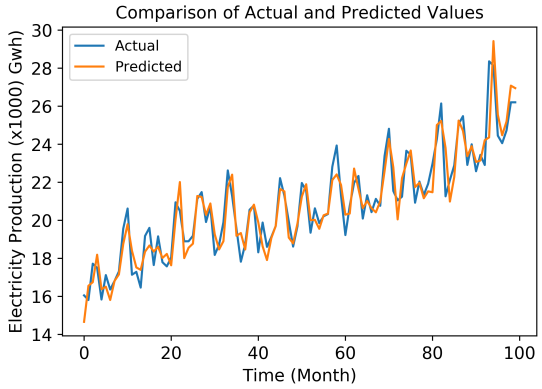

Figure 14. Comparison of actual values and predicted values by MLP.

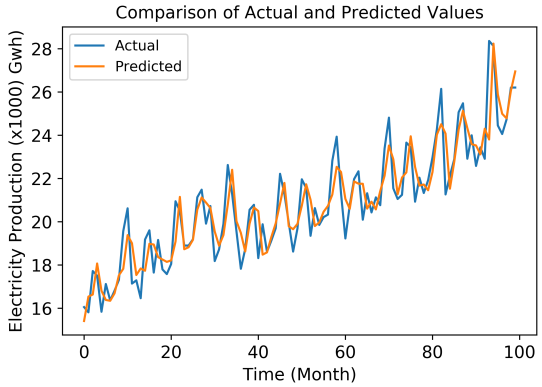

Figure 15. Comparison of actual values and predicted values by SVR.

4. Discussion and Conclusion

In this study, we have focused on how to convert a sequential dataset to a dataset to which a supervised learning algorithm can be applied. Also, we have compared three different methods in terms of various evaluation metrics which are commonly used in regression problems. Based on the results given in Section 3, the deep learning method LSTM outperforms the other two approaches in terms of all evaluation metrics in the majority of the number of lags. Table 5 shows the average performance of all methods regarding chosen evaluation metrics

Table 5. The average error rates of three methods across all lags.

Lags	MAE	MSE (1000s)	RMSE	R^2
MLP	1090.13	2022.45	1381.62	0.86
LSTM	855.543	1279.50	1104.68	0.91
SVR	971.28	1599.09	1251.54	0.88

Through the number of lags, the MLP method fluctuates too much so that yields the worst performance in general. On the other hand, LSTM is the most successful method algorithm among the others it is more robust to the number of lags compare to MLP method. Although SVR cannot outperform the LSTM method in the majority of the lags, it still gives acceptable results when using the high number of lags. Table 6 illustrates how many times each algorithm gives better results than others for all given evaluation metrics. For example, LSTM method outperforms others by giving the best performance for all metrics 14 times, MLP 5 times, and SVR 1 time. One needs to note that, this finding is only for the best performance comparison, so it is not possible to infer SVR is the worst method. If we look at the comparison between SVR and MLP, we can conclude the SVR outperform MLP method regarding the average evaluation scores as shown in Table 7.

Table 6. Number of the best performance of each algorithm.

Method	MLP	SVR	LSTM
The best performance	5	1	14

Table 7. Number of the best performance of SVR and MLP algorithms.

Method	MLP	SVR
The best performance	8	12

As details given in Section 3, a too small and too big number of lags highly alter the performance of methods because of the lack of enough information or meaningful information from the past of the data. For future research, finding a way to choose the optimum number of lags before setting up the experiment can save from computational time and might yield optimum performance. Also, we have proved that deep learning algorithm is the best candidate for future forecasting to be able to get the prediction with the minimum error.

References

[1] Zafer Dilaver and Lester C Hunt. Industrial electricity demand for turkey: a structural time series analysis. *Energy Economics*, 33(3):426–436, 2011.

[2] Alper Ünler. Improvement of energy demand forecasts using swarm intelligence: The case of turkey with projections to 2025. *Energy Policy*, 36(6):1937–1944, 2008.

[3] M Duran Toksarı. Estimating the net electricity energy generation and demand using the ant colony optimization approach: case of turkey. *Energy Policy*, 37(3):1181–1187, 2009.

[4] Yi Lin, Mian-yun Chen, and Sifeng Liu. Theory of grey systems: capturing uncertainties of grey information. *Kybernetes*, 33(2):196–218, 2004.

[5] Diyar Akay and Mehmet Atak. Grey prediction with rolling mechanism for electricity demand forecasting of turkey. *Energy*, 32(9):1670–1675, 2007.

[6] Coskun Hamzacebi and Huseyin Avni Es. Forecasting the annual electricity consumption of turkey using an optimized grey model. *Energy*, 70:165–171, 2014.

[7] Volkan Ş Ediger and Sertac Akar. Arima forecasting of primary energy demand by fuel in turkey. *Energy Policy*, 35(3):1701–1708, 2007.

[8] Erkan Erdogdu. Natural gas demand in turkey. *Applied Energy*, 87(1):211–219, 2010.

[9] Jun-song Jia, Jing-zhu Zhao, Hong-bing Deng, and Jing Duan. Ecological footprint simulation and prediction by arima modelâ€"a case study in henan province of china. *Ecological Indicators*, 10(2):538–544, 2010.

[10] Ali Sait Albayrak. Arima forecasting of primary energy production and consumption in turkey: 1923-2006. *Enerji, piyasa ve düzenleme*, 1(1):24–50, 2010.

[11] Samuel Asuamah Yeboah, Manu Ohene, TB Wereko, et al. Forecasting aggregate and disaggregate energy consumption using arima models: a literature survey. *Journal of Statistical and Econometric Methods*, 1(2):71–79, 2012.

[12] Kadir Kavaklioglu. Modeling and prediction of turkeyâ€™s electricity consumption using support vector regression. *Applied Energy*, 88(1):368–375, 2011.

[13] Yetis Sazi Murat and Halim Ceylan. Use of artificial neural networks for transport energy demand modeling. *Energy policy*, 34(17):3165–3172, 2006.

[14] Adnan Sozen, Erol Arcaklioglu, and Mehmet Ozkaymak. Modelling of turkey's net energy consumption using artificial neural network. *International Journal of Computer Applications in Technology*, 22(2-3):130–136, 2005.

[15] Serhat Kucukali and Kemal Baris. Turkeyâ€™s short-term gross annual electricity demand forecast by fuzzy logic approach. *Energy policy*, 38(5):2438–2445, 2010.

[16] Coşkun Hamzaçebi. Forecasting of turkey's net electricity energy consumption on sectoral bases. *Energy policy*, 35(3):2009–2016, 2007.

[17] Ujjwal Kumar and VK Jain. Time series models (grey-markov, grey model with rolling mechanism and singular spectrum analysis) to forecast energy consumption in india. *Energy*, 35(4):1709–1716, 2010.

[18] Ramazan Ünlü and Petros Xanthopoulos. Estimating the number of clusters in a dataset via consensus clustering. *Expert Systems with Applications*, 125:33–39, 2019.

[19] Ramazan Ünlü and Petros Xanthopoulos. A weighted framework for unsupervised ensemble learning based on internal quality measures. *Annals of Operations Research*, 276(1-2):229–247, 2019.

[20] Amanpreet Singh, Narina Thakur, and Aakanksha Sharma. A review of supervised machine learning algorithms. In *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 1310–1315. IEEE, 2016.

[21] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[22] Yuming Hua, Junhai Guo, and Hua Zhao. Deep belief networks and deep learning. In *Proceedings of 2015 International Conference on Intelligent Computing and Internet of Things*, pages 1–4. IEEE, 2015.

[23] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[24] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[25] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*, 2010.

[26] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE, 2013.

[27] Nal Kalchbrenner and Phil Blunsom. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, 2013.

[28] Vladimir Vapnik. *Statistical learning theory. 1998*, volume 3. Wiley, New York, 1998.

[29] Harris Drucker, Christopher JC Burges, Linda Kaufman, Alex J Smola, and Vladimir Vapnik. Support vector regression machines. In *Advances in neural information processing systems*, pages 155–161, 1997.

[30] Richard Lippmann. An introduction to computing with neural nets. *IEEE Assp magazine*, 4(2):4–22, 1987.

[31] Teuvo Kohonen. *Self-organization and associative memory*, volume 8. Springer Science & Business Media, 2012.

[32] David E Rumelhart and James L McClelland. Parallel distributed processing: explorations in the microstructure of cognition. volume 1. foundations. 1986.

[33] Teuvo Kohonen. An introduction to neural computing. *Neural networks*, 1(1):3–16, 1988.

[34] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[35] Jiawei Han, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.

[36] Ian H Witten, Eibe Frank, Mark A Hall, and Christopher J Pal. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.