Computers & Industrial Engineering 143 (2020) 106435

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

An optimized model using LSTM network for demand forecasting

Hossein Abbasimehr[a,*], Mostafa Shabani[b], Mohsen Yousefi[c]

[a] Faculty of Information Technology and Computer Engineering, Azarbaijan Shahid Madani University, Tabriz, Iran
[b] IT Group, Department of Industrial Engineering, KN Toosi University of Technology, Tehran, Iran
[c] Demand Planning and Logistic Department, NILPER FURNITURE, Tehran, Iran

ARTICLE INFO

Keywords:
Demand prediction
Time series forecasting
Statistical methods
LSTM
RNN

ABSTRACT

In a business environment with strict competition among firms, accurate demand forecasting is not straightforward. In this paper, a forecasting method is proposed, which has a strong capability of predicting highly fluctuating demand data. Therefore, in this paper we propose a demand forecasting method based on multi-layer LSTM networks. The proposed method automatically selects the best forecasting model by considering different combinations of LSTM hyperparameters for a given time series using the grid search method. It has the ability to capture nonlinear patterns in time series data, while considering the inherent characteristics of non-stationary time series data. The proposed method is compared with some well-known time series forecasting techniques from both statistical and computational intelligence methods using demand data of a furniture company. These methods include autoregressive integrated moving average (ARIMA), exponential smoothing (ETS), artificial neural network (ANN), K-nearest neighbors (KNN), recurrent neural network (RNN), support vector machines (SVM) and single layer LSTM. The experimental results indicate that the proposed method is superior among the tested methods in terms of performance measures.

1. Introduction

Demand forecasting is the basis of all planning activities (Haberleitner, Meyr, & Taudes, 2010). Specifically, demand prediction as a predictive analytics task is considered an essential tool to gain an understanding of future demand. Accurate demand forecasting guarantees suitable supply chain management, and enhances customer satisfaction by preventing inventory stock-out (Kumar, Shankar, & Alijohani, 2019).

Strict competition among firms in any domain has made it difficult for businesses to accurately forecast the customers' demands using traditional demand forecasting methods (Guo, Diao, Zhao, Wang, & Sun, 2017; Kumar et al., 2019). Therefore, companies are increasingly moving toward the use of advanced data science techniques to forecast customer demand. In general, customer demand is modeled as a sequential data of customer demands over time. Hence, demand forecasting problem can be formulated as a time series forecasting problem (Villegas, Pedregal, & Trapero, 2018). Time series prediction has been applied in various areas of application such as credit scoring (Lin, Hu, & Tsai, 2011), electricity load forecasting (Johannesen, Kolhe, & Goodwin, 2019; Raza, Nadarajah, & Ekanayake, 2017), forecasting call center arriving calls (Taylor, 2008) tourism demand forecasting (Law, Li, Fong, & Han, 2019), ATM cash demand forecasting (Martínez, Frías, Pérez, & Rivera, 2017), forecasting of petroleum production (Sagheer & Kotb, 2019), weather forecasting (Maqsood, Khan, & Abraham, 2004), etc.

Generally, time series forecasting techniques fall into the two main categories of statistical and computational intelligence methods (Khashei & Bijari, 2011). Widely-used statistical time series forecasting methods such as ARIMA suppose that the time series contains only linear components. However, most real-world time series data consist of nonlinear components too. To address forecasting of time series with nonlinear patterns, several nonlinear statistical methods have been developed; for example, the autoregressive conditional heteroscedastic (ARCH) model, general autoregressive conditional heteroscedastic (GARCH) (Khashei & Bijari, 2011). However, there are many variations of these models (Enders, 2008), each suitable at modeling only a specific nonlinearity. This causes the procedure of finding a proper model for time series to become more complex.

Recently, computational intelligence techniques including artificial neural networks (ANN), support vector machine (SVM), K-nearest neighbors (KNN), and adaptive neuro-fuzzy inference system (ANFIS) have been frequently used for the problem of time series prediction.

ANNs have several advantageous characteristics, including universal approximation, being data driven, and the ability to better capture nonlinear patterns in data (Khashei & Bijari, 2011; Panigrahi &

* Corresponding author.
E-mail addresses: abbasimehr@azaruniv.ac.ir (H. Abbasimehr), mshabani@mail.kntu.ac.ir (M. Shabani), m.yousefi@nilper.ir (M. Yousefi).

https://doi.org/10.1016/j.cie.2020.106435
Received 21 July 2019; Received in revised form 16 March 2020; Accepted 22 March 2020
Available online 30 March 2020
0360-8352/

Behera, 2017). A specific class of ANNs are recurrent neural networks (RNNs). Unlike in feedforward ANNs, the connections between nodes in an RNN establish a cycle which allows signals to move in different directions (Parmezan, Souza, & Batista, 2019). RNNs provide a short-term memory by storing the activations from each time step. This makes it a suitable technique for processing sequence data (Parmezan et al., 2019). The weakness of RNNs is the vanishing and exploding gradient problem, which makes it hard to train (Bengio, Simard, & Frasconi, 1994; Parmezan et al., 2019). The prevalent solution to overcome this weakness is to use gated architectures such as LSTM (Hochreiter & Schmidhuber, 1997), which can exploit longer-range timing information (Wu, Yuan, Dong, Lin, & Liu, 2018; Xin, Ji, & Chao, 2018).

Although different types of ANNs can capture nonlinear patterns of time series data, research have indicated that the ANNs with shallow architectures are unable to accurately model time series with a high degree of nonlinearity, longer range and heterogeneous characteristics (Sagheer & Kotb, 2019; Taieb, Bontempi, Atiya, & Sorjamaa, 2012). In addition, it is demonstrated that deep neural network architectures have better generalization than shallow architectures (Hermans & Schrauwen, 2013; Utgoff & Stracuzzi, 2002).

In this study, we propose a method based on a multi-layer LSTM network by using the grid search approach. The proposed method searches for the optimal hyperparameters of the LSTM network. The capability to capture nonlinear patterns in time series data is one of the main advantages of our method. We apply the proposed method on real-world demand data of a furniture company, and compare it to other state-of the-art time series forecasting techniques. The results of these methods are compared, and we demonstrate that the model built by employing the proposed method performs substantially better than the alternatives.

The rest of this paper is organized as follows. Section 2 gives a comprehensive literature review on time series forecasting along with a brief description of methods utilized throughout this paper. In Section 3, we describe the proposed method. Section 4 presents a case study using the proposed method as well as the methods used for comparison. Also, this section analyzes and compares the results of utilized methods. In Section 5, we draw a conclusion and suggest future works.

2. Related work

In this section, we firstly present a comprehensive literature review on time series forecasting and identify the utilized techniques and the context of each study. Afterward, the utilized forecasting techniques throughout this paper are described.

Time series forecasting has been applied in many areas of application. Table 1 summarizes prevalent research in the context of time series forecasting, that have been published during the past decade. The table highlights contribution, modeling methods and techniques, the domain of study, and the superiority and drawback of each research. As the table indicates, many modeling techniques has been employed in those researches. These modeling techniques primarily come from statistical and computational intelligence fields. The statistical methods ARIMA and seasonal ARIMA (Babu & Reddy, 2014; de Oliveira & Ludermir, 2016; Khandelwal, Adhikari, & Verma, 2015; Khashei & Bijari, 2011; Lemke & Gabrys, 2010; Liu, Tian, & Li, 2012; Murray, Agard, & Barajas, 2018; Panigrahi & Behera, 2017; Parmezan et al., 2019; Raza et al., 2017; Sagheer & Kotb, 2019; Sarıca, Eğrioğlu, & Aşıkgil, 2018; Yang, Chen, Wang, Li, & Li, 2016), moving average (Andrawis, Atiya, & El-Shishiny, 2011; Lemke & Gabrys, 2010; Parmezan et al., 2019), and exponential smoothing (de Oliveira & Ludermir, 2016; Lemke & Gabrys, 2010; Panigrahi & Behera, 2017; Parmezan et al., 2019; Sarıca et al., 2018) are widely-used. Also from the computational intelligence field, ANN (Andrawis et al., 2011; Atsalakis, 2016; Babu & Reddy, 2014; Khandelwal et al., 2015; Khashei & Bijari, 2011; Lemke & Gabrys, 2010; Panigrahi & Behera, 2017; Parmezan et al., 2019; Raza et al., 2017; Sarıca et al., 2018; Yang et al., 2016) is a widely-used method. In addition, other computational intelligence techniques including KNN (Martínez, Frías, Pérez-Godoy, & Rivera, 2018; Martínez et al., 2017; Parmezan et al., 2019), ANFIS (Atsalakis, 2016; Atsalakis, Protopapadakis, & Valavanis, 2016; Yang et al., 2016), SVM (Chen, Yang, Liu, Li, & Li, 2015; de Oliveira & Ludermir, 2016; Parmezan et al., 2019) and LSTM (Parmezan et al., 2019; Sagheer & Kotb, 2019; Shi, Hu, & Zhang, 2019) have been employed for time series forecasting.

Furthermore, some hybrid techniques have been developed to take advantages of both statistical and computational intelligence techniques (Babu & Reddy, 2014; de Oliveira & Ludermir, 2016; Khandelwal et al., 2015; Khashei & Bijari, 2011).

In this paper we propose a demand forecasting method based on multi-layer LSTM networks. Our proposed method effectively considers temporal dependencies in time series. The proposed method automatically selects the best forecasting model by considering different combinations of LSTM hyperparameters for a given time series by using the grid search method. It has the ability to capture nonlinear patterns in time series data, while considering the inherent characteristics of non-stationary time series data. Using a furniture company's demand data, the proposed method is compared with time series prediction techniques from the statistical and computational intelligent method classes. The results show that the proposed method is superior to the other methods, and proved that the proposed method has practical significance and innovation.

2.1. Utilized models

In this section we describe briefly the applied models in this study.

2.1.1. ARIMA

ARIMA (Box, Jenkins, Reinsel, & Ljung, 2015) comprises three procedures: autoregression, integration and moving average (Parmezan et al., 2019). ARIMA can perform modeling of different kinds of time series (Ramos, Santos, & Rebelo, 2015). The main limitation of ARIMA is that it assumes that the given time series is linear (Adhikari & Agrawal, 2013).

2.1.2. Exponential smoothing

Exponential smoothing methods are similar to moving average, except for the fact that predictions are obtained by considering weighted averages of past values in a time series (Hyndman, Koehler, Ord, & Snyder, 2008). Fifteen different extensions of exponential methods have been developed. These methods are classified based on trend and seasonality. For each of fifteen models, two possible innovation state space models were described by (Hyndman et al., 2008). A triplet (E: error, T: trend, S: seasonality) is used for distinguish the thirty models (Hyndman et al., 2008).

2.1.3. KNN

K-nearest neighbors (KNN) regression is a machine learning technique that has recently been successfully applied for time series forecasting (Martínez et al., 2018; Martínez et al., 2017). The main idea of applying KNN for time series forecasting is straightforward; given a series $T = (t_1, t_2, \cdots, t_n)$, the problem is to forecast T_{n+h}, where h is the forecast horizon. To apply KNN, a set of instances are created from time series T. The target of an instance is a historical value of $T$and the input features describing the instance are lagged values (for example lag $= l$) of the target. This set of instances is considered the train set. For predicting an instance Q with target t_{n+1}, which contains $(t_{n-l}, \ldots, t_{n-1}, t_n)$ as input feature, the KNN regression method searches for the k most similar instances to$Q$. When the $k$most similar instances are found, then target of Q is obtained by averaging the targets of the $k$found instances.

2.1.4. ANN

ANN is a computational model that emulates human brain

Table 1
Literature review.

Reference	Contribution	Methods and techniques	Domain	Superiority (S)/Drawback (D)
(Lemke & Gabrys, 2010)	Proposing a *meta*-learning for model selection in time series forecasting context – i.e. achieving knowledge on which time series model should be selected for a given forecasting task.	Moving average, Single exponential smoothing, Taylor smoothing, Theta-model, Polynomial regression, ARIMA, ANN, Elman NN.	NN3 and NN5 cash demand datasets.	**S**: Giving insights into selecting a suitable forecasting technique for time series forecasting. **D**: The results are only based on NN3 and NN5 datasets.
(Andrawis et al., 2011)	Proposing a combined forecasting model by averaging the forecasts of individual predictors. Based on the previous studies, several forecasting models were tested to be incorporated in the combined model.	Gaussian process regression, ANN, Gaussian process regression, Moving average.	NN5 cash demand forecast	**S**: Proposing a new method to deal with the seasonality of data; another superiority is conducting a large number of experiments to select the best models to combine them. **D**: Lack of validation procedures to select right models.
(Khashei & Bijari, 2011)	Proposing a novel hybrid model of ARIMA and ANN that yields more accurate forecasting than traditional combined methods.	ARIMA, ANN, and ARIMA-ANN	Wolf's sunspot data, the Canadian lynx data, and the British pound/US dollar exchange rate data	**S**: Taking advantages of both linear models and non-linear models in time series prediction. **D**: The proposed model may give poor results in cases when the original time series contains only non-linear components.
(Babu & Reddy, 2014)	Proposing a hybrid ARIMA-ANN model by exploring the volatility of data employing a moving-average filter.	ARIMA, ANN, ARIMA-ANN	Sunspot data, electricity price data, and stock market data	**S**: Proposing a hybrid ARIMA-ANN that differs from previously proposed ARIMA-ANN as it diagnoses whether a given time series is normally distributed or not by computing the kurtosis of the time series. In other words, it firstly decomposes the time series into low-volatile and high-volatile components and then use ARIMA for modelling low-volatile and ANN for modelling high volatile components. **D**: Decomposition of time series causes more data to be lost.
(Chen et al., 2015)	Proposing a hybrid forecasting method that firstly filters noisy data and then removes the seasonal pattern and finally forecasts the preprocessed data using the least squares support vector machine model.	Least squares support vector machine	Load forecasting	**S**: Considering the internal characteristics of the time series data and proposing a preprocessing approach based on the empirical mode decomposition to filter noise from data. **D**: The utilized methods in experiments are SVM-based method and there is not any statistical or ANN-based method.
(de Oliveira & Ludermir, 2016)	Proposing a new ARIMA-SVR model for time series forecasting which considers the nature of a time series. Firstly, a simple ETS filter is applied to decompose series into low and high volatility components. The low volatility component is modeled using ARIMA and the high volatility part is modeled by AR-SVR.	ARIMA, ETS, SVR	In this study, 12 datasets from the Time Series Data Library (Hyndman & Akram, 2010) and one electricity price data from the Australian National Electricity Market (Babu & Reddy) were used to assess the performance of the methods.	**S**: Optimizing parameters of ARIMA and SVR using particle swarm optimization (PSO). **D**: The drawback is that decomposition of time series using exponential smoothing filter may make more data to be lost.
(George S Atsalakis, 2016)	Utilizing three computational intelligence methods to forecast changes in carbon price	ANN, ANFIS, and a hybrid neuro-fuzzy controller system (PATSOS)	Daily carbon emissions futures prices from October 14, 2009 to October 29, 2013. This yields a sample of 1074 observations.	**S**: Extending the application of neuro-fuzzy controller to carbon price prediction. Also, making extensive comparisons with 13 time series forecasting methods to demonstrate the performance of the proposed method. **D**: Lack of explanations on how to choose the number of membership functions for the input features.
(Raza et al., 2017)	Proposing an ensemble framework for time series forecasting. The framework consists of wavelet transform, forecasting techniques, PSO, and Bayesian model averaging.	ARIMA, Elman neural network, feed-forward neural network (FNN), backpropagation neural network (BPNN) radial basis function (RBF).	Load demand forecast	**S**: Using Bayesian model averaging (BMA) to combine the output of the individual forecasters instead of considering equal weights to the models. Also, incorporating several various forecasters in the ensemble is another advantage of this study. **D**: Lack of presenting an approach for selecting individual forecasters in the ensemble framework.

(*continued on next page*)

Table 1 (*continued*)

Reference	Contribution	Methods and techniques	Domain	Superiority (S)/Drawback (D)
(Panigrahi & Behera, 2017)	Proposing a hybrid Exponential time series smoothing (ETS)-ANN model in order to forecast linear and non-linear time series.	ETS, ARIMA, ANN, ETS-ANN	Sixteen datasets from the Time Series Data Library	**S**: The proposed model is able to handle linear, non-linear time series and also time series containing of a combination of linear and non-linear components. **D**: Deep learning methods are not employed.
(Martínez et al., 2017)	Proposing a methodology for using KNN algorithm in time series by exploring different preprocessing and modelling approaches	KNN	ATM cash forecasting	**S**: Giving new insights about the impact of different preprocessing and modelling techniques on the performance of KNN when utilized on NN3 competition time series.**D**: The main drawback of this paper is that its findings are based on evaluating models only on NN3 data set.
(Murray et al., 2018)	Proposing a methodology based on existing forecasting techniques to predict demands of individual customers via segmenting customers and using the segment-level forecast to obtain individual demand.	ARIMA	Supplier of bulk liquid materials. The dataset consists ofdelivery records for approximately five years.	**S**: Providing a method to segment customers and use ARIMA to obtain segment-level forecast and finally to apply the obtained model to forecast demand of individual customers.**D**: The main drawback is that Computational intelligence methods are not used for demand forecasting.
(Martínez et al., 2018)	Proposing a new method for addressing the seasonality of time series in time series prediction using KNN	KNN	ATM cash forecasting	**S**: Improving KNN performance when applied on NN5 dataset by handling seasonality through narrowing training set.**D**: The performance of the method is not evaluated on other time series data.
(Sarıca et al., 2018)	Proposing a method through combining autoregressive (AR) and adaptive network fuzzy inference system (ANFIS)	Exponential Smoothing, SRIMA, FNN, ANFIS.	12 time series data including one time series related to Australian Beer Consumption (ABC), five time series from Istanbul stock exchange market, and six time series related to Taiwan stock exchange capitalization weighted stock index (TAIEX)	**S**: Taking the advantage of neuro-fuzzy methods and ARIMA.**D**: Lack of explanation on how the number of membership functions (the number of clusters of Fuzzy C-Means) are chosen.
(Sagheer & Kotb, 2019)	Proposing a deep LSTM architecture for time series forecasting of two oilfields	ARIMA, Deep Gated Recurrent Unit (DGRU), RNN, Deep LSTM, Nonlinear Extension for linear Arps decline (NEA), Higher-Order Neural Network (HONN)	Petroleum industry	**S**: Utilizing a wide range of methods including statistical, computational intelligence, and deep learning methods for petroleum production forecasting. **D**: The main weakness of the proposed methodology is smoothing the raw data with the aim of reducing noise. So, the raw data is replaced with smoothed data which can be straightforwardly forecasted.
(Parmezan et al., 2019)	Performing extensive experiments using eleven forecasting techniques from both parametric and non-parametric models on 95 datasets and proposing some recommendations regarding the best situation each model can be employed	ARIMA, SARIMA, Moving Average, Simple Exponential Smoothing, Holt's exponential smoothing, Holt-Winters' seasonal exponential smoothing (Multiplicative and Additive), ANN, SVM, KNN, LSTM	95 datasets including 40 synthetic datasets and 55 real datasets (agriculture, climatology, engineering, finance, physics, tourism.	**S**: Conducting a comprehensive experimental comparison to provide a better insight into the predictive power of the commonly-used time series forecasting methods. **D**: Lack of including the deep learning-based methods.
(Shi et al., 2019)	Proposing a double hidden layer LSTM for forecasting gyroscope shell temperature time series data	ANN, SVM, LSTM	Gyroscope shell temperature data	**S**: Using deep learning-based method to forecast the temperature of the gyroscope. **D**: The drawback is that the statistical methods such as ARIMA are not employed. Also, the statistical characteristics of the time series is not analyzed before applying forecasting methods.

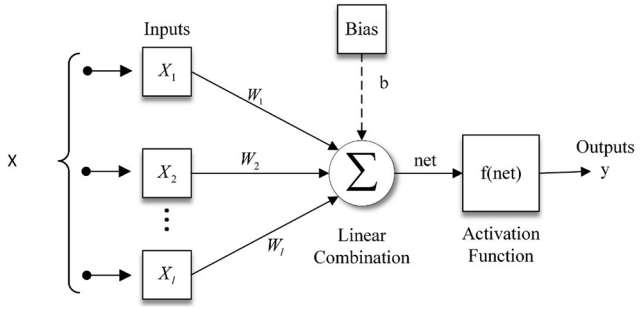

Fig. 1. The structure of a perceptron.

operations in processing information. ANNs are nonparametric and data-driven, so model misspecification impacts the performance less than they would for parametric models (Khashei & Bijari, 2011). ANNs with one hidden layer are one of the extensively used types of ANNs for time series forecasting (Khashei & Bijari, 2011). The ANN depicted in Fig. 1 is the simplest form of ANN and is called a perceptron.

In Fig. 1, the single neuron comprises p data inputs $x_i \in X$. The i^{th} element of X, is associated with a weight w_i. This weight indicates the importance of the i^{th}input. The net value is computed by summing weighted inputs and a bias $b \in \Re$. The output $y$of the neuron is computed by sending net value to an activation function f. The output y can be binary or continuous. The weights of the perceptron are adjusted by performing learning process during a determined number of iterations.

One common type of ANN with more than two layers is multilayer perceptron (MLP) which is trained by applying backpropagation learning algorithm (Rumelhart et al., 1985). The performance of MLP models is dependent on the number of hidden layer, the number of neurons in each layer, and the weights among neurons in different layers (Parmezan et al., 2019).

2.1.5. Simple RNN

Recurrent neural networks are different from MLPs in that in RNNs connections between neurons form a directed graph including some self-reflective connections (Parmezan et al., 2019). RNNs can model temporal dependencies and are especially suitable for the prediction of sequence data (Parmezan et al., 2019).

2.1.6. LSTM

LSTM as an extension of RNN has a strong capability in forecasting time series data. The main difference between an RNN and LSTM is that LSTM can store long-range time dependency information and can suitably map between input and output data (Greff, Srivastava, Koutník, Steunebrink, & Schmidhuber, 2017). The LSTM network structure differs from the conventional perceptron architecture as it contains a cell and gates which controls the flow of information. Specifically, the LSTM contains an input gate, a forget gate, internal state (cell memory), and an output gate (Greff et al., 2017) as illustrated in Fig. 2. The notations are as follows.

$x(t_i)$: The input value

$h(t_{i-1})$ and $h(t_i)$: The output value at time point t_{i-1} and t_i

$c(t_{i-1})$ and $c(t_i)$: Cell states at time points t_{i-1} and t_i

$b = \{b_a, b_f, b_c, b_o\}$ are biases of input gate, forget gate, internal state and output gate.

$\vec{W_1} = \{w_a, w_f, w_c, w_o\}$ are weight matrixes of input gate, forget gate, internal state and output gate.

$\vec{W_2} = \{w_{ha}, w_{hf}, w_{hc}, w_{ho}\}$ are the recurrent weights

$\vec{a} = \{a(t_i), f(t_i), c(t_i), o(t_i)\}$ are the output results for input gate, forget gate, internal state, and output gate.

Considering these notations, the operation of LSTM is as follows:

The forget gate $f(t_i)$ uses $x(t_i)$ and $h(t_{i-1})$ as input to compute the information to be preserved in $c(t_{i-1})$ using a sigmoid activation.

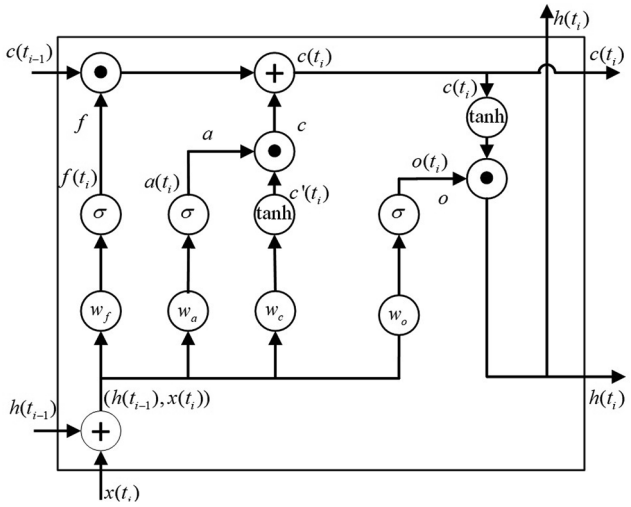

Fig. 2. The structure of LSTM (Greff et al., 2017).

The input gate $a(t_i)$ takes $x(t_i)$ and $h(t_{i-1})$ to compute the value of $c(t_i)$.

The output gate $o(t_i)$ performs regulation on the output of an LSTM cell by considering $c(t_i)$ and applying both sigmoid and tanh layers. Mathematically the forward learning of an LSTM is as follows:

$$a(t_i) = \sigma(w_a x(t_i) + w_{ha} h(t_{i-1}) + b_a) \quad (1)$$

$$f(t_i) = \sigma(w_f x(t_i) + w_{hf} h(t_{i-1}) + b_f) \quad (2)$$

$$c(t_i) = f_t \times c(t_{i-1}) + a_t \times \tanh(w_c x(t_i) + w_{hc}(h(t_{i-1}) + b_c) \quad (3)$$

$$o(t_i) = \sigma(w_o x(t_i) + w_{ho} h(t_{i-1}) + b_o) \quad (4)$$

$$h(t_i) = o(t_i) \times \tanh(c(t_i)) \quad (5)$$

where σ and tanh are activation functions, and $\times$ indicates point-wise multiplication.

Overall, the LSTM learns using the following steps:

(1) Compute the LSTM output using Eqs. (1)–(5) (forward learning).
(2) Compute the error between the resulted data and input data of each layer.
(3) The error is reversely propagated to input gate, cell, and forget gate.
(4) Based on the error term, the weight of each gate is updated using an optimization algorithm.

The above four-step process are repeated for a given number of iteration and the optimal values of weights and biases are obtained (Greff et al., 2017).

2.1.7. Support vector machines (SVM)

SVM is a machine learning method which is based on the statistical learning theory used for solving classification and regression tasks (Han, Kamber, & Pei, 2011). Assuming a set of data points from two classes, an SVM chooses a hyperplane that separates the data points with maximal margin. For data that are not linearly separable, SVM uses kernels such as linear, polynomial, and Radial Basis Function (RBF) to map the data points into a higher-dimensional space in which they become separable (Han et al., 2011).

3. The proposed methodology

The aim of this study is to obtain an accurate model for demand forecasting of a furniture company. We exploit recent deep learning methods to specify the best time series forecasting model for solving the demand forecasting problem. The proposed methodology for sale time

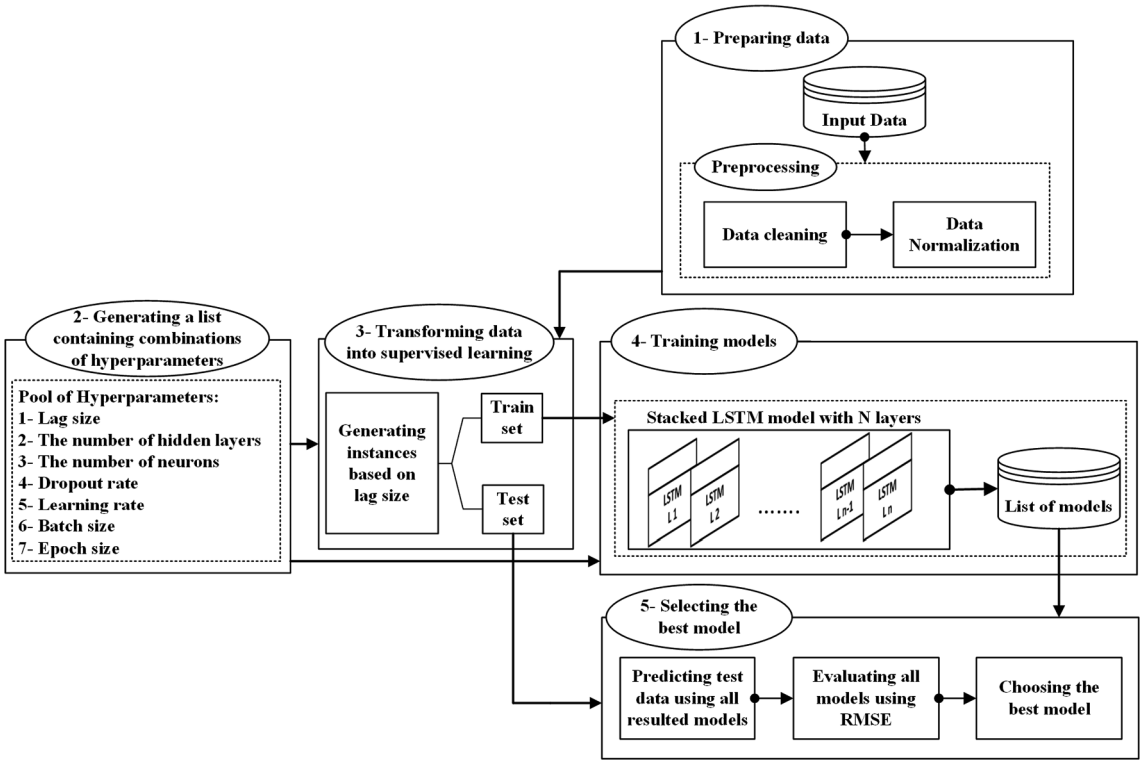

Fig. 3. The proposed methodology for sale forecasting using time series forecasting approach.

series forecasting is illustrated in Fig. 3 which describes the steps of the proposed method. Trying to overcome the challenges of obtaining an accurate forecasting model and considering the intrinsic characteristics of the demand time series (being nonlinear and non-stationary), the proposed method focuses mainly on optimization of hyperparameters of LSTM network. The steps of the methodology are described next.

3.1. Preparing data

Given a time series $s$with length N $\{s(t_i), i = 1, 2,...,N\}$ the time series firstly should be preprocessed. Data cleaning and normalization are the two essential data preparation tasks which will be applied to prepare data for forecasting task.

3.1.1. Data cleaning

In case the time series contains noisy and missing values, the noisy values are smoothed and the missing values are replaced using an appropriate technique.

3.1.2. Data Normalization:

The time series $s(t_i)$ is normalized and the resulted normal data expressed as $\{s_n(t_i), i = 1, 2,...,N\}$

$$s_n(t_i) = \frac{s(t_i) - \min(s)}{\max(s) - \min(s)} \tag{6}$$

where $\min(s)$ is the minimum value of s and $\max(s)$ is the maximum value of s.

3.2. Generating a list containing combinations of hyperparameters for LSTM

LSTM network has achieved acceptable performance when applied on sequence data (Reimers & Gurevych, 2017). However, obtaining good performance with LSTM networks is not a simple task, as it involves the optimization of multiple hyperparameters (Reimers & Gurevych, 2017). For time series forecasting problems, the important hyperparameters are the lag size (i.e. number of past observations), the number of hidden layers, the dropout rate, the number of neurons in each layer, the optimization algorithm, the learning rate, the epoch size, and the batch size. Hyperparameter selection improve the model performance (Reimers & Gurevych, 2017). In this study, we have used the grid search method which is a widely-used method to search optimal parameters (Reimers & Gurevych, 2017). For our experiments we adjusted seven different hyperparameters as shown in Fig. 3. The parameter setting using grid search is as follows:

Firstly, we set predefined values for each parameter. Then a list of parameter combinations is generated. As indicated in the Fig. 3, the list of hyperparameters for grid search purpose are:

1. **Lag size:** The lag size parameter has significant impact on the performance of time series forecasting (Ribeiro et al., 2011). Therefore, it is crucial to test the performance of a model using different lag sizes.
2. **The number of hidden layers**: It is shown that deep neural network architectures have better generalization than single layer architectures (Hermans & Schrauwen, 2013; Utgoff & Stracuzzi, 2002). As a result, it is important to investigate the performance of a model with more than one layer.
3. **The number of neurons:** Selecting the optimal number of neurons for each layer in the LSTM network is not a straightforward task. If the number of neurons is very small, the LSTM will not able to memorize all necessary information to perform prediction optimally. Also, if the number of neurons is very high, the LSTM will overfit on the training instances and will not demonstrate suitable generalization to accurately forecast test set (Reimers & Gurevych, 2017).
4. **Dropout rate:** Applying dropout rate on LSTM influences the performance of the resulted model by enhancing the model generalization (Reimers & Gurevych, 2017; Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014).
5. **Learning rate:** Learning rate is an important hyperparameter which adjusts the extent of change to the model weights. Choosing the best

learning rate is essential in obtaining a model with high performance (Reimers & Gurevych, 2017).

6. **Batch size**: Adjusting batch size is another factor in determining the performance of the LSTM model. Hence it is significant to find an optimal value for batch size (Shi et al., 2019).
7. **Epoch size**: One training epoch is referred to a single iteration over all training instances (Reimers & Gurevych, 2017). If the number of training epochs is too small, the model will not capture the patterns of training instances. Also, if the epoch number is too large, the model will suffer from overfitting. Therefore, finding a suitable epoch number is vital in achieving a model with high performance.

Depending on the number of values for each hyperparameters, a list of parameter combinations is generated and utilized for data transformation along with model configuration and training.

3.3. Transforming data into supervised learning

In this step, firstly time series data are reshaped into a set of instances with predefined input features and an output feature. Then the created instances are split into train and test sets.

3.3.1. Generating instances based on lag size

Based on the lag size selected in the previous step, the time series is converted into a set of input and output format. This data reshaping task is required to employ an LSTM. Using the lag L, the instances are created as follows:

$$s_{input} = [s_1, s_2, ..., s_{N-L}]^T = \begin{bmatrix} s_n(t_1) & s_n(t_2) & \cdots & s_n(t_L) \\ s_n(t_2) & s_n(t_3) & \cdots & s_n(t_{L+1}) \\ \cdots & \cdots & \cdots & \cdots \\ s_n(t_{N-L}) & s_n(t_{N-L+1}) & & s_n(t_{N-1}) \end{bmatrix} \quad (7)$$

The corresponding output value can be denoted as follows:

$$s_{output} = [o_1, o_2, ..., o_{N-L}] = \begin{bmatrix} s_n(t_{L+1}) \\ s_n(t_{L+2}) \\ \cdots \\ s_n(t_N) \end{bmatrix} \quad (8)$$

3.3.2. Splitting instances into train and test sets

In this step, the instances are divided into two separate parts including the train and test sets. The train set is input to the LSTM network to obtain forecasting model. The test set is used for evaluating the predictive power of the built model in terms of performance metrics.

3.4. Model configuration and training

As our proposed method is mainly based on exploiting multi-layer LSTM; so here, we firstly describe the structure of a stacked LSTM.

3.4.1. Specification of stacked LSTM

In the stacked LSTM as illustrated in Fig. 3, the input at time t_i, $s(t_i)$ is input into the first LSTM hidden layer along with the $h^{(1)}(t_{i-1})$, the output value of the first LSTM block at time point t_{i-1}. The output value of first block at time t_i is computed which is $h^{(1)}(t_i)$ The second hidden layer uses the output value $h^{(1)}(t_i)$ along with the previous output value of the second block, $h^{(2)}(t_{i-1})$, to calculate the output value of the second LSTM $h^{(2)}(t_i)$; that in turn $h^{(2)}(t_i)$ becomes the input value to the third hidden layer along with the previous output value of the third block, $h^{(3)}(t_{i-1})$ and this process continues until the final LSTM in the model which calculate the output value. The predominant advantage of the stacked LSTM structure is that each layer can model some part of the time series and then pass on to the next block until finally the last accumulated block computes the output.

3.4.2. Modeling

Except for the lag size parameter, the remaining hyperparameters including number of hidden layers, epoch size, batch size, the number of neurons, learning rate and dropout rate are used in configuring and training the LSTM network. Having determined the values of these parameters using the grid search method in the previous step, in this step, corresponding to each hyperparameter combination, a LSTM network is created and trained using train set.

3.5. Selecting the best model

In this step, to evaluate the generated candidate models, performance of each model is assessed using RMSE metric which is calculated based on denormalized input and predicted output data. Afterward, the best model is selected.

4. Experimental setup and results

To evaluate the performance of the proposed approach, two statistical time series forecasting methods (ETS (Hyndman et al., 2008), and ARIMA (Box et al., 2015)), and five computational intelligence methods (ANN, SVM, KNN, simple RNN, and the single layer LSTM) were considered. The ETS and ARIMA models were executed and optimized using Statsmodels[1] package in Python. The SVM and KNN were implemented and optimized using scikit-learn package[2] in python. Also ANN, RNN, LSTM were implemented using Keras (Chollet, 2015) in Python.

For evaluation of all models we use RMSE and SMAPE performance measures (Martínez et al., 2018). SMAPE is defined by:

$$SMAPE = \frac{1}{n}\sum_{t=1}^{n}\frac{|\hat{y}_t - y_t|}{\frac{|\hat{y}_t| + |y_t|}{2}} \quad (9)$$

RMSE is expressed as:

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{y}_t)^2} \quad (10)$$

In both (9) and (10), y_t and $\hat{y}_t$ are the actual and forecast values of the series in time point t respectively.

4.1. Data description

The employed data constitutes the monthly sale quantity for a product from 2007 to 2017 of a furniture company. The time series of sale quantity consists of 132 months in total.

In general, managing the material and product available is a key aspect of every industry. The best way to manage the optimum quantity of raw material and product is to find out the future demand for these materials and products. To reach this goal, the businesses must predict the future demands based on their previous sales. The data used in this paper is the sale data of a furniture company. This data is the monthly sale quantity for a specific product from 2007 to 2017. The company seeks to accurately predict the demand for its products in order to effectively plan for supplying raw materials. The prediction of future demand based on this data can help the company to improve its strategies for resource management. As the analyzed data is for a long period of time, the resulted demand prediction can be reliable and cover the future demand behavior of market.

4.1.1. Characteristics of the time series

One of the important tasks in time series analysis is to determine the degree of stationarity of a given time series. The mean and variance of a

[1] https://www.statsmodels.org/stable/index.html.
[2] https://scikit-learn.org/stable/.

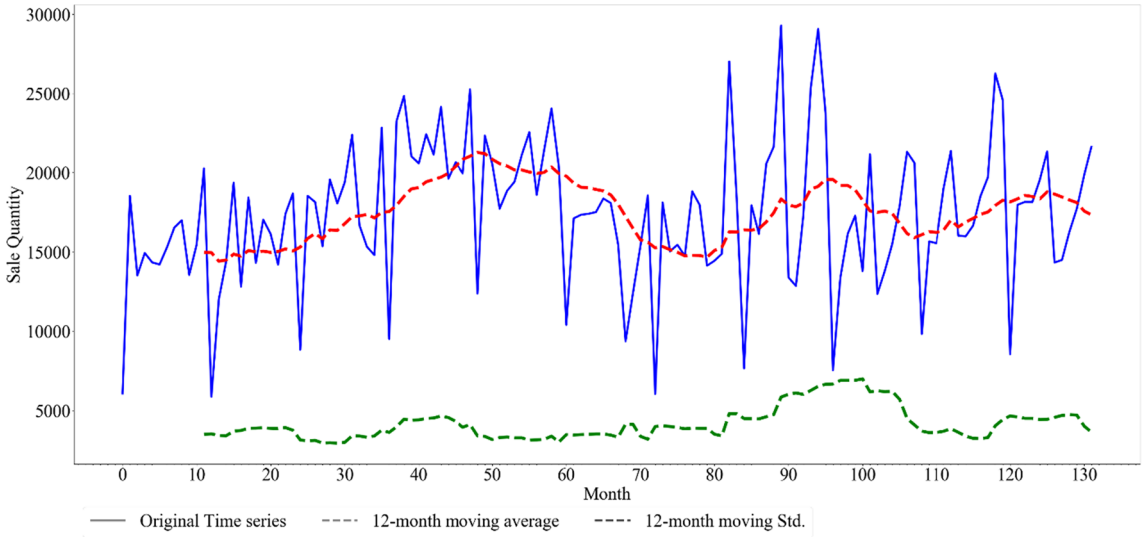

Fig. 4. Trend plot of the time series.

stationary time series remain constant in time. Trends and seasonality are two common components of time series that lead to non-stationarity. Fig. 4 exhibits the time series' row values, 12-month moving averages, and 12-month rolling standard deviations. The figure demonstrates that the time series has both trend and seasonality. Therefore, it is essential to measure the degree of non-stationarity.

To measure the degree of stationarity of the time series, we used the Augmented Dickey Fuller (ADF) test (Engle & Yoo, 1987). The p-value and ADF statistics of applying the ADF test on the time series are given in Table 2. Here, the p - value>0.05, and therefore, we cannot reject the null hypothesis (H0) and the time series has a unit root and is non-stationary.

The trend plot of the time series illustrated in Fig. 4 reveals several significant characteristics of the time series. The series starts with an uptrend which reverses to a downtrend; and this cycle is repeated for the rest of its duration. These sudden shifts in trend make the forecasting task difficult. Another feature that is apparent throughout the time series is non-constant variance (heteroskedasticity). This means that the variability of time series in different rolling windows is unequal. This characteristic is captured in the 12-month standard deviation plot (green line) (as seen in Fig. 4).

4.1.2. Investigation of the predictability of the time series

The most important characteristic of a chaotic system is the unpredictability because of the sensitive dependence on initial conditions (Yang & Wu, 2010). In a chaotic system, the divergence will be exponential in time. According to chaos theory, Lyapunov exponents are an indicator of a system's predictability. The maximal Lyapunov exponent (MLE) is a technique to detect the presence of chaotic behavior. The MLE indicates the averaged information of the divergence of a system and thus determines its predictability (Yang, Chen, & Jiang, 2015). Negative Lyapunov exponents show convergence, while positive Lyapunov exponents indicate divergence and chaos (McCue & Troesch, 2011). Several algorithms have been proposed to calculate the maximal Lyapunov exponent (Ghorbani, Kisi, & Aalinezhad, 2010). Rosenstein, Collins, and De Luca (1993) developed an algorithm to calculate the MLE from a given time series which is a widely used algorithm. In general, the existing algorithms to calculate Lyapunov exponents require stationary data (Salisbury & Sun, 2004). However, we showed that the utilized time series is non-stationary. Therefore, to analyze chaos of the time series and obtain the MLE, it is essential to decompose the non-stationary time series into stationary subseries.

The main idea of the empirical mode decomposition (EMD) (Huang, Shen, Long, Wu, Shih, Zheng, & Liu, 1998) is to decompose a time series data into a set of subseries that are called intrinsic mode functions (IMFs). These IMFs have two important characteristics: Each IMFs has its own time scale characteristics, and IMFs are relatively stationary (Büyükşahin & Ertekin, 2019). To decompose the time series into IMFs, we exploited the PyEMD Library (Laszuk, 2017). The obtained IMFs are firstly checked for stationarity. To check the stationarity of each IMF, we employed the Augmented Dickey Fuller (ADF) test (Engle & Yoo, 1987). The results of the ADF test of the IMFs indicated that all of them are stationarity. Therefore, we applied the chaos analysis to all IMFs. We used the Rosenstein et al. (1993) algorithm to obtain the maximal Lyapunov exponent and the results are demonstrated in Table 3. As seen in Table 3, IMF 1, IMF 2, IMF 3, and IMF 4 have a positive Lyapunov exponent and thus are chaotic. According to (Salisbury & Sun, 2004), since four IMFs are chaotic, the original time series must also be chaotic. According to the Eq. (11), The inverse of the maximal Lyapunov exponent defines the predictability time (forecast horizon) for a chaotic time series (Khatibi et al., 2012) is defined by:

$$F_H = \left\lfloor \frac{1}{MLE} \right\rfloor \tag{11}$$

A larger MLE indicates shorter forecast horizons. The forecast horizons (F_H) of each IMF is illustrated in Table 3.

As the results indicated, the original time series must be chaotic, therefore, it cannot be predicted for an arbitrary length. However, the

Table 2
ADF test results on the time series.

	Value
ADF Statistic	−2.6883
p-value	0.4609
Critical Value 1%	−4.472
Critical Value 5%	−3.883
Critical Value 10%	−3.585

Table 3
The maximal Lyapunov exponents corresponding to the extracted IMFs.

IMF	MLE	F_H
IMF 1	0.0056	178
IMF 2	0.0157	63
IMF 3	0.0064	153
IMF 4	0.0096	104
IMF 5	−0.014	

Lyapunov exponents corresponding to the IMFs are low and consequently, the forecast horizons of the IMFs are long. Although we cannot certainly determine the maximal predictability of the original time series from its IMFs, the original demand series can be predicted over a longer forecast horizon (for example, for a 12-month horizon) using ANNs. It should be noted that the accuracy of predictions will decline with longer forecast horizons. In this study, we set the forecast horizon equal to 20% of time-series data.

4.2. Preparing data

The given time series has no missing value. Also, in order to preserve the characteristics of the real-world data, no noise reduction or data smoothing was performed on the series. Since most of the forecasting techniques presents more promising forecasting ability by using normalized data, in this study, we normalized the data using min-max normalization algorithm (Han et al., 2011).

4.3. Generating a list containing combinations of hyperparameters

In this step, for each hyperparameter, a list of values is determined to be used in model configuration and training. The search space of each hyperparameters is illustrated in the Table 4.

4.4. Transforming data into supervised learning

Based on the lag size selected using the grid search method, the time series is transformed into a set of instances with input-output format. For instance, for lag size = 10, a subset of the created instances is shown in Table 5. Then, the created instances are split into a train and test set. For all experiments, the test set contains 20% of all instances.

4.5. Training models

For each combination of hyperparameters, an LSTM network is designed and trained. For all networks, the Adam algorithm (Kingma & Ba, 2014) was used as optimizer. In addition, we set the mean of squared error (MSE) as loss function for all designed LSTM networks.

4.6. Selecting the best model

The obtained models in the previous step are evaluated using RMSE to choose the best one. The hyperparameters for the best obtained model is illustrated in Table 6.

4.7. Comparison with the widely-used techniques

According to (Khashei & Bijari, 2011), there is no dominant model in the time series forecasting. Our literature review demonstrated that a plethora of forecasting techniques have been used in past researches. However, none of the forecasting techniques has shown performance that is superior to the others in general. Therefore, to compare the forecasting power of our model, it is compared with some state-of-the-art-models, including ARIMA, exponential smoothing, KNN, ANN, RNN and LSTM. It is important to note that for each benchmark method, we have searched for optimal parameters on which the particular method performs best. The optimized parameters of the benchmark methods are depicted in Table 7.

Table 4
Values specified for each hyperparameter.

Hyperparameter	Values
Lag	[10, 11, 12, 13]
The number of hidden layers	[2, 3, 4]
The number of neurons in each layer	[4, 8, 16, 32, 64,128]
Dropout rate	[0.05, 0.1, 0.15, 0.2, 0.25]
Learning rate	[0.01, 0.001, 0.0001]
Batch size	[1, 5, 10, 20]
Epoch size	[50,100, 200, 300, 400, 500]

The performance of applying the proposed method along with the benchmark methods on the demand time series data are illustrated in Table 8. Notably, the results shown in Table 8 indicate the performance of the corresponding method on both train and test set. However, we provide our analysis of the results using the performances on test set. From

Table 8, it is clear that the best results achieved using our approach in terms of RMSE and SMAPE.

In the next subsections, point-wise comparisons between our proposed method and each benchmark method is accomplished.

4.8. Our method versus statistical methods

ETS and ARIMA are the widely applied statistical method for time series forecasting. Here, we compare the performance of our proposed method with them. The actual and forecasted data using our proposed method, ETS and ARIMA is illustrated in Fig. 5.

The proposed method outperforms ETS in terms of both performance measures. In terms of SMAPE, ETS performs better than methods such as ARIMA, RNN, and KNN. The reason for this result is that ETS can handle both linear and non-linear patterns, whereas ARIMA cannot (Panigrahi & Behera, 2017).

ARIMA can accurately forecast time series that only contain linear components. As seen from Table 7, the proposed method performs better than ARIMA. This demonstrates that the time series data used in this case study contains nonlinear patterns that ARIMA has a problem capturing.

4.9. Our method versus computational intelligence methods

In this section, we compare the performance of our proposed method with the applied computational intelligence methods. Fig. 6 illustrates the actual and forecasted data using our proposed method, and the computational intelligences methods including SVM, ANN, LSTM, RNN and KNN.

4.9.1. Our method versus KNN

KNN method has been applied successfully for time series prediction (Martínez et al., 2018). However, its performance is not appropriate for our data in terms of RMSE and SMAPE. Regarding RMSE and SMAPE, its performance is lower than other contenders.

4.9.2. Our method versus RNN

RNN seems to be more suitable method for time series forecasting. However, it suffers from vanishing and exploding gradient problem (Sagheer & Kotb, 2019). Applied on our data, RNN outperformed KNN. However, it achieved worse performance in comparison to the other methods in terms of both RMSE and SMAPE.

4.9.3. Our method versus single layer LSTM

As described before, the best optimal configuration for our method is using double hidden LSTM layer. As indicated in Table 8, our method has superior performance than LSTM in terms of both RMSE and SMAPE. This is due the fact that containing double LSTM layer improve the generalization of our method. Besides, our method utilized dropout process which furtherly enhance the model generalization.

4.9.4. Our method versus ANN

Although our method outperforms ANN, there is no substantial difference between these two methods in terms of SMAPE. This demonstrates the power of ANN in modeling real data with high nonlinearity. Also, as observable from Table 8, ANN achieved better models

Table 5
A subset of instances created.

	Input instances with lag size 10										Output
	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}
Ins1	0.2624	0.3659	0.5768	0.2967	0.5368	0.3610	0.4769	0.4383	0.3559	0.4929	0.5475
Ins2	0.3659	0.5768	0.2967	0.5368	0.3610	0.4769	0.4383	0.3559	0.4929	0.5475	0.1269
Ins3	0.5768	0.2967	0.5368	0.3610	0.4769	0.4383	0.3559	0.4929	0.5475	0.1269	0.5411
Ins4	0.2967	0.5368	0.3610	0.4769	0.4383	0.3559	0.4929	0.5475	0.1269	0.5411	0.5244
Ins5	0.5368	0.3610	0.4769	0.4383	0.3559	0.4929	0.5475	0.1269	0.5411	0.5244	0.4055
Ins6	0.3610	0.4769	0.4383	0.3559	0.4929	0.5475	0.1269	0.5411	0.5244	0.4055	0.5851

Table 6
The hyperparameters for the best resulted LSTM model.

Hyperparameter	Value
Lag	12
The number of hidden layers	2
The number of neurons in each layer	64
Dropout rate	0.1
Learning rate	0.001
Batch size	1
Epoch size	500

Table 7
The best parameters of the obtained models.

Algorithm	The best Parameters
ETS	Trend: Additive Seasonality: Additive Length of one seasonal period (s):12
ARIMA	Length of one seasonal period (s):12 Order and seasonal Order: The best parameters obtained using the Auto-ARIMA function.
KNN	Lag:14 Number of nearest neighbors (k):3
SVM	Lag:14 Regularization parameter (C):100 Kernel: Radial Basis Function (RBF)
ANN	Lag:12 Number of hidden layers = 1 Number of units in the hidden layer (n):64 Learning rate (ϒ) :0.001 Number of epochs (E): 85
RNN	Lag:12 Number of hidden layers = 1 Number of units in the hidden layer (n):64 Learning rate (ϒ) :0.01 Number of epochs (E): 500
LSTM	Lag:12 Number of hidden layers = 1 Number of units in the hidden layer (n):64 Learning rate (ϒ) :0.001 Number of epochs (E): 146

Table 8
SMAPE and RMSE of All methods.

Method	Train		Test			
	SMAPE	RMSE	SMAPE	RMSE	Rank SMAPE	Rank RMSE
ETS	0.1316	3121.28	0.1212	2669.49	5	3
ARIMA	0.1616	3531.78	0.1222	2869.41	6	5
KNN	0.1374	2913.9	0.1540	3392.18	8	8
Simple RNN	0.1333	2788.4	0.1379	3040.5	7	7
LSTM	0.1072	2267.11	0.1208	2961.52	4	6
ANN	**0.1011**	2115.65	0.1138	2664.76	3	2
SVM	0.1101	**1967.8**	0.1122	2706.90	2	4
Our method	0.1023	2172.4	**0.1085**	**2595.96**	**1**	**1**

than other methods except SVM and our method in terms of SMAPE. Furthermore, in terms of RMSE, ANN has better output than SVM. Specifically, ANN achieved 2664.76 RMSE whereas SVM achieved 2706.90.

4.9.5. Our method versus SVM

From Table 8, it is clear that our method outperforms SVM both in terms of RMSE and SMAPE. In fact, SVM is the second-best performing technique among all other methods in terms of SMAPE. The results of this study demonstrate good performance of SVM in time series forecasting. The main advantage of SVM over the other neural network methods we tested is that SVM has a small number of parameters and finding optimal parameters of this method is easier than ANN.

4.10. Statistical comparisons of the applied models

To assess whether the performance of the proposed approach is significantly better than the performances of the other methods, it is required to carry out statistical tests (Protopapadakis, Niklis, Doumpos, Doulamis, & Zopounidis, 2019). To perform statistical tests, we created four extra time series from the original time series. The four time series datasets $\{D_1, D_2, D_3, D_4\}$ are created as follows: for obtaining $D_1$the last $h$points of the original time series are selected as test set and the remaining part is considered as training set. Similarly, for obtaining D_2, we firstly remove the h previously selected points from the original time series. Then the last $h$points of the new original time series are selected as the test set and the remaining time points as training set. This procedure is repeated for obtaining D_3, D_4. Considering that the original time series is the monthly demand data, we set $h = 12$ for creating the datasets. Also, in the previous section we have created a dataset by splitting the time series into a training set with 105 time points and the test set with 27 time points. Hereafter, we name that dataset as $D_5$and use it in our statistical comparison. After creating the four datasets, we applied the top six methods on each time series dataset and obtained the best performing models corresponding to each method. In the following, we report the performance only in terms of SMAPE as it is popular accuracy measure in the context of time series forecasting. Table 9 depicts the SMAPE values achieved by each applied method.

In this paper, the two-step technique presented by Demšar (2006) is employed. Based on that procedure, at first, the Friedman's test is conducted to detect whether there are significant differences among the applied models concerning SMAPE.

The null hypothesis for the Friedman test is that there are no differences among results. If the null hypothesis is rejected, then we proceed with Hochberg's post hoc tests (Demšar, 2006) to determine whether the proposed method is significantly better than the other applied methods. In the following, we describe the procedure to conduct the Friedman test:

1- Based on the performance results of the applied models, for each dataset $i \in [1, D]$ we rank values from 1 (corresponding to the lowest SMAPE) to k (corresponding to the highest SMAPE). These ranks are denoted as $r_i^j (j \in [1, k])$. In this paper, we have$k = 6$

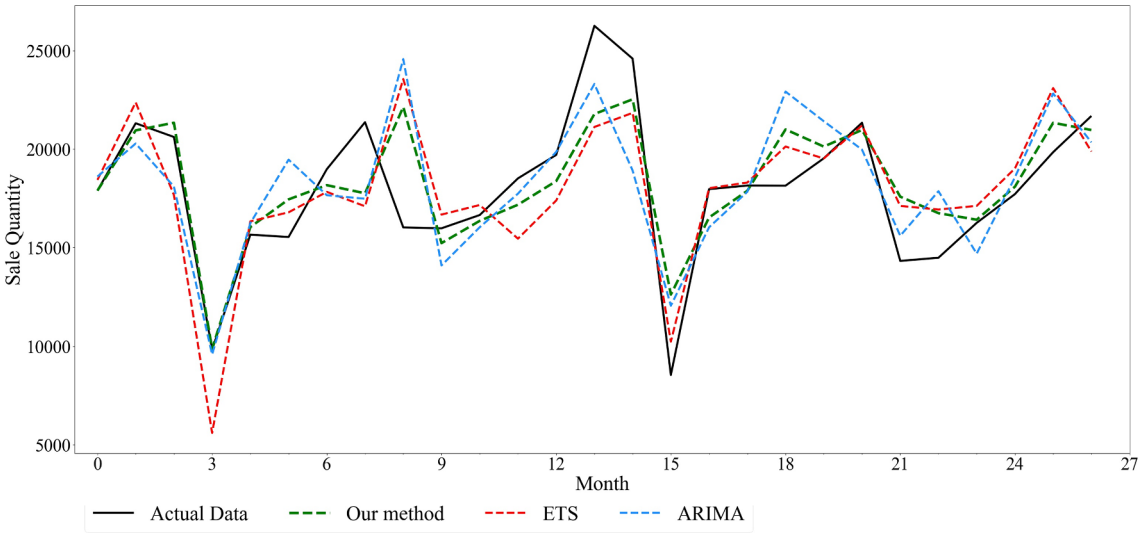

Fig. 5. Actual data vs prediction using our method and the statistical methods (ETS and ARIMA).

forecasting methods that are compared in $D = 5$ datasets (time series).

2- For each method j, compute its average rank obtained in all datasets -$AR_j = \frac{1}{D}\sum_{i=1}^{D} r_i^j$ where r_i^j denotes the rank of the method j on dataset i.

The Friedman test compares the average ranks, AR_j of different methods on the utilized datasets. Under the null hypothesis, which states that all methods perform similarly, the Friedman statistic is computed by Eq. (12)

$$\chi_F^2 = \frac{12D}{k(k+1)}\left[\sum_{j=1}^{k} AR_j^2 - \frac{k(k+1)^2}{4}\right] \tag{12}$$

where χ_F^2 is the chi-squared distribution with $k-1$ degrees of freedom. If the value of χ_F^2 is large enough, then the null hypothesis is rejected (Demšar, 2006). Table 10 illustrated ranks calculated through the Friedman test.

The obtained Friedman statistic ($\chi_F^2 = 9.46$) is greater than the critical value at the 0.05 level ($\chi_F^2 = 9.067$) (Sheskin, 2003) which indicates that the null hypothesis is rejected with significant level 0.05. Therefore, there are significant differences among the applied models.

Table 9
The performance results of all methods on 5 datasets in terms of SMAPE.

Applied methods	D1	D2	D3	D4	D5
ARIMA	0.1257	0.1310	0.1966	0.2535	0.1222
ETS	0.0829	0.1704	**0.1631**	0.2630	0.1212
SVM	0.1259	**0.1169**	0.2182	0.2816	0.1122
ANN	0.1020	0.1299	0.2712	0.3629	0.1138
LSTM	0.1127	0.1441	0.2765	0.3235	0.1208
Proposed method	**0.0730**	0.1295	0.1801	**0.2513**	**0.1085**

Table 10
Friedman ranks.

Applied methods	D1	D2	D3	D4	D5	AR
ARIMA	5	4	3	2	6	4
ETS	2	6	**1**	3	5	3.4
SVM	6	**1**	4	4	2	3.4
ANN	3	3	5	6	3	4
LSTM	4	5	6	5	4	4.8
Proposed method	**1**	2	2	**1**	**1**	**1.4**

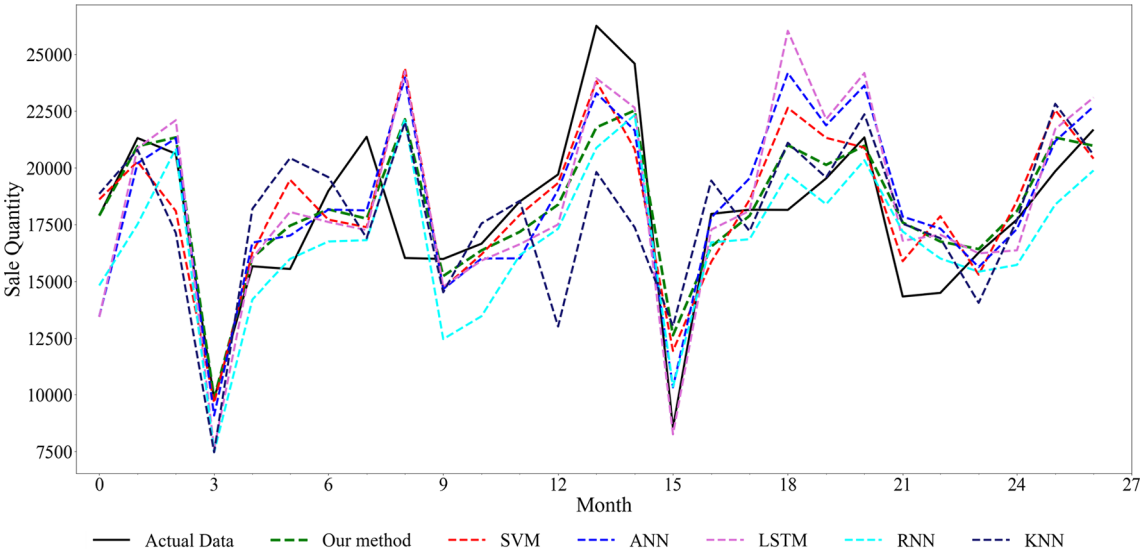

Fig. 6. Actual data vs prediction using our method and the computational intelligence methods.

Table 11
P-values of post-hoc tests comparing the combined method with other individual forecasters.

Applied models	ARIMA	ETS	SVM	ANN	LSTM
z	**2.1974**	**1.6903**	**1.6903**	**2.1974**	**2.8735**
***p*-value**	0.027992	0.090971	0.090971	0.027992	0.004066

In the second step, Hochberg's post hoc test (Demšar, 2006) was used to test whether any significant differences between individual forecasters. In fact, this test is used for comparing the proposed method $i$and other method j by computing the z using Eq. (13):

$$z = \frac{(R_i - R_j)}{\sqrt{\frac{k(k+1)}{6D}}} \tag{13}$$

In Hochberg's test, the p-values of the z for $k-1$ comparisons with the control model (the proposed method) are firstly computed and then sorted from the largest to the smallest. Next, a series of pairwise comparisons is carried out as follows: firstly, the largest p-value is compared with significance level $\alpha = 0.1$; then the second largest p-value is compared with $\alpha/2$ and so forth, until a moment where a rejection of the null hypothesis is found. All the remaining hypotheses are rejected too.

Table 11 illustrates the p-values corresponding to the z-value in Eq. (13), computed when comparing the average rank of the proposed method with the other methods. It is apparent that there are significant differences in the average ranks between the proposed method and the other methods. In fact, the largest p-value is 0.090971 which is lower than $\alpha = 0.1$. Therefore, the corresponding null hypothesis is rejected. Also, all hypotheses with smaller p-values are then rejected as well. This indicates that there are significant differences in the average rank between the proposed method and the other applied methods.

5. Conclusion

In this study, we propose a multilayer LSTM network for demand forecasting. The proposed method has the ability to configure an LSTM network which can effectively model patterns of a time series. We compare the proposed method with some well-known time series forecasting techniques from both the statistical and computational intelligence methods categories. To determine whether the performance of the proposed approach is significantly better than the performances of the other methods, statistical tests are carried out. The results demonstrate the superiority of the proposed method in compression to the all utilized method. Furthermore, our experiments on the original demand time series reveal that our method reaches the best performance in terms of both RMSE and SMAPE, followed by SVM and ANN. This indicates that computational intelligence techniques have great capability in capturing patterns of real-world time series data. Besides, the results of the experiments showed that traditional statistical methods such as ETS and ARIMA yield weak performance in comparison to SVM, ANN, and LSTM. The obtained model is employed for forecasting future demand. As future work, we can employ the proposed method on other areas such as customer future behavior forecasting. Also, the future study can be utilizing other deep learning methods such as attention-based neural networks for time series forecasting.

CRediT authorship contribution statement

Hossein Abbasimehr: Conceptualization, Methodology, Writing - original draft, Writing - review & editing. **Mostafa Shabani:** Software, Writing - review & editing. **Mohsen Yousefi:** Resources, Writing - review & editing.

Acknowledgement

The authors would like to thank Lukas Hedegaard Jensen for providing language help during the writing of this paper.

References

Adhikari, R., & Agrawal, R. K. (2013). An introductory study on time series modeling and forecasting. arXiv preprint arXiv:1302.6613.
Andrawis, R. R., Atiya, A. F., & El-Shishiny, H. (2011). Forecast combinations of computational intelligence and linear models for the NN5 time series forecasting competition. *International Journal of Forecasting, 27*(3), 672–688.
Atsalakis, G. S. (2016). Using computational intelligence to forecast carbon prices. *Applied Soft Computing, 43*, 107–116.
Atsalakis, G. S., Protopapadakis, E. E., & Valavanis, K. P. (2016). Stock trend forecasting in turbulent market periods using neuro-fuzzy systems. *Operational Research, 16*(2), 245–269. https://doi.org/10.1007/s12351-015-0197-6.
Babu, C. N., & Reddy, B. E. (2014). A moving-average filter based hybrid ARIMA–ANN model for forecasting time series data. *Applied Soft Computing, 23*, 27–38.
Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks, 5*(2), 157–166.
Box, G. E., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). *Time series analysis: Forecasting and control.* New Jersey: John Wiley & Sons.
Büyükşahin, Ü.Ç., & Ertekin, Ş. (2019). Improving forecasting accuracy of time series data using a new ARIMA-ANN hybrid method and empirical mode decomposition. *Neurocomputing, 361*, 151–163. https://doi.org/10.1016/j.neucom.2019.05.099.
Chen, Y., Yang, Y., Liu, C., Li, C., & Li, L. (2015). A hybrid application algorithm based on the support vector machine and artificial intelligence: An example of electric load forecasting. *Applied Mathematical Modelling, 39*(9), 2617–2632.
Chollet, F. (2015). Keras.
de Oliveira, J. F., & Ludermir, T. B. (2016). A hybrid evolutionary decomposition system for time series forecasting. *Neurocomputing, 180*, 27–34.
Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research, 7*, 1–30.
Enders, W. (2008). *Applied econometric time series.* John Wiley & Sons.
Engle, R. F., & Yoo, B. S. (1987). Forecasting and testing in co-integrated systems. *Journal of econometrics, 35*(1), 143–159.
Ghorbani, M. A., Kisi, O., & Aalinezhad, M. (2010). A probe into the chaotic nature of daily streamflow time series by correlation dimension and largest Lyapunov methods. *Applied Mathematical Modelling, 34*(12), 4050–4057. https://doi.org/10.1016/j.apm.2010.03.036.
Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., & Schmidhuber, J. (2017). LSTM: A Search Space Odyssey. *IEEE Transactions on Neural Networks and Learning Systems, 28*(10), 2222–2232. https://doi.org/10.1109/TNNLS.2016.2582924.
Guo, F., Diao, J., Zhao, Q., Wang, D., & Sun, Q. (2017). A double-level combination approach for demand forecasting of repairable airplane spare parts based on turnover data. *Computers & Industrial Engineering, 110*, 92–108. https://doi.org/10.1016/j.cie.2017.05.002.
Haberleitner, H., Meyr, H., & Taudes, A. (2010). Implementation of a demand planning system using advance order information. *International Journal of Production Economics, 128*(2), 518–526.
Han, J., Kamber, M., & Pei, J. (2011). *Data Mining: Concepts and Techniques: Concepts and Techniques.* Waltham, USA: Elsevier Science.
Hermans, M., & Schrauwen, B. (2013). Training and analysing deep recurrent neural networks. Paper presented at the Advances in neural information processing systems.
Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735–1780.
Huang, N. E., Shen, Z., Long, S. R., Wu, M. C., Shih, H. H., Zheng, Q., . . . Liu, H. H. (1998). The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis. Proceedings of the Royal Society of London. Series A: mathematical, physical and engineering sciences, 454(1971), 903-995.
Hyndman, R., Koehler, A. B., Ord, J. K., & Snyder, R. D. (2008). *Forecasting with exponential smoothing: The state space approach.* Springer Science & Business Media.
Hyndman, R. J., & Akram, M. (2010). Time series data library. Available from Internet: http://robjhyndman. com/TSDL.
Johannesen, N. J., Kolhe, M., & Goodwin, M. (2019). Relative evaluation of regression tools for urban area electrical energy demand forecasting. *Journal of Cleaner Production, 218*, 555–564.
Khandelwal, I., Adhikari, R., & Verma, G. (2015). Time series forecasting using hybrid ARIMA and ANN models based on DWT decomposition. *Procedia Computer Science, 48*, 173–179.
Khashei, M., & Bijari, M. (2011). A novel hybridization of artificial neural networks and ARIMA models for time series forecasting. *Applied Soft Computing, 11*(2), 2664–2675.
Khatibi, R., Sivakumar, B., Ghorbani, M. A., Kisi, O., Koçak, K., & Farsadi Zadeh, D. (2012). Investigating chaos in river stage and discharge time series. *Journal of Hydrology, 414–415*, 108–117. https://doi.org/10.1016/j.jhydrol.2011.10.026.
Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.
Kumar, A., Shankar, R., & Alijohani, N. (2019). A big data driven framework for demand-driven forecasting with effects of marketing-mix variables. *Industrial Marketing Management.* https://doi.org/10.1016/j.indmarman.2019.05.003.
Laszuk, D. (2017). Python implementation of Empirical Mode Decomposition algorithm. Retrieved from. http://www.laszukdawid.com/codes.
Law, R., Li, G., Fong, D. K. C., & Han, X. (2019). Tourism demand forecasting: A deep

learning approach. *Annals of Tourism Research, 75*, 410–423.
Lemke, C., & Gabrys, B. (2010). Meta-learning for time series forecasting and forecast combination. *Neurocomputing, 73*(10–12), 2006–2016.
Lin, W.-Y., Hu, Y.-H., & Tsai, C.-F. (2011). Machine learning in financial crisis prediction: A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 42*(4), 421–436.
Liu, H., Tian, H.-Q., & Li, Y.-F. (2012). Comparison of two new ARIMA-ANN and ARIMA-Kalman hybrid methods for wind speed prediction. *Applied Energy, 98*, 415–424.
Maqsood, I., Khan, M. R., & Abraham, A. (2004). An ensemble of neural networks for weather forecasting. *Neural Computing & Applications, 13*(2), 112–122.
Martínez, F., Frías, M. P., Pérez-Godoy, M. D., & Rivera, A. J. (2018). Dealing with seasonality by narrowing the training set in time series forecasting with kNN. *Expert Systems with Applications, 103*, 38–48.
Martínez, F., Frías, M. P., Pérez, M. D., & Rivera, A. J. (2017). A methodology for applying k-nearest neighbor to time series forecasting. *Artificial Intelligence Review*, 1–19.
McCue, L. S., & Troesch, A. W. (2011). *Use of Lyapunov exponents to predict chaotic vessel motions Contemporary Ideas on Ship Stability and Capsizing in Waves*. Springer415–432.
Murray, P. W., Agard, B., & Barajas, M. A. (2018). Forecast of individual customer's demand from a large and noisy dataset. *Computers & Industrial Engineering, 118*, 33–43.
Panigrahi, S., & Behera, H. S. (2017). A hybrid ETS–ANN model for time series forecasting. *Engineering Applications of Artificial Intelligence, 66*, 49–59.
Parmezan, A. R. S., Souza, V. M., & Batista, G. E. (2019). Evaluation of statistical and machine learning models for time series prediction: Identifying the state-of-the-art and the best conditions for the use of each model. *Information Sciences, 484*, 302–337.
Protopapadakis, E., Niklis, D., Doumpos, M., Doulamis, A., & Zopounidis, C. (2019). Sample selection algorithms for credit risk modelling through data mining techniques. *International Journal of Data Mining, Modelling and Management, 11*(2), 103–128.
Ramos, P., Santos, N., & Rebelo, R. (2015). Performance of state space and ARIMA models for consumer retail sales forecasting. *Robotics and Computer-Integrated Manufacturing, 34*, 151–163.
Raza, M. Q., Nadarajah, M., & Ekanayake, C. (2017). Demand forecast of PV integrated bioclimatic buildings using ensemble framework. *Applied Energy, 208*, 1626–1638.
Reimers, N., & Gurevych, I. (2017). Optimal hyperparameters for deep lstm-networks for sequence labeling tasks. arXiv preprint arXiv:1707.06799.
Ribeiro, G. H. T., Neto, P. S. G. d. M., Cavalcanti, G. D. C., & Tsang, I. R. (2011, 31 July-5 Aug. 2011). Lag selection for time series forecasting using Particle Swarm Optimization. Paper presented at the The 2011 International Joint Conference on Neural Networks.
Rosenstein, M. T., Collins, J. J., & De Luca, C. J. (1993). A practical method for calculating largest Lyapunov exponents from small data sets. *Physica D: Nonlinear Phenomena, 65*(1–2), 117–134.
Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1985). Learning internal representations by error propagation: California Univ San Diego La Jolla Inst for Cognitive Science.
Sagheer, A., & Kotb, M. (2019). Time series forecasting of petroleum production using deep LSTM recurrent networks. *Neurocomputing, 323*, 203–213.
Salisbury, J. I., & Sun, Y. (2004). Assessment of chaotic parameters in nonstationary electrocardiograms by use of empirical mode decomposition. *Annals of Biomedical Engineering, 32*(10), 1348–1354.
Sarıca, B., Eğrioğlu, E., & Aşıkgil, B. (2018). A new hybrid method for time series forecasting: AR–ANFIS. *Neural Computing and Applications, 29*(3), 749–760.
Sheskin, D. J. (2003). *Handbook of parametric and nonparametric statistical procedures*. Chapman and Hall/CRC.
Shi, H., Hu, S., & Zhang, J. (2019). LSTM based prediction algorithm and abnormal change detection for temperature in aerospace gyroscope shell. *International Journal of Intelligent Computing and Cybernetics, 12*(2), 274–291.
Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research, 15*(1), 1929–1958.
Taieb, S. B., Bontempi, G., Atiya, A. F., & Sorjamaa, A. (2012). A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition. *Expert Systems with Applications, 39*(8), 7067–7083.
Taylor, J. W. (2008). A comparison of univariate time series methods for forecasting intraday arrivals at a call center. *Management Science, 54*(2), 253–265.
Utgoff, P. E., & Stracuzzi, D. J. (2002). Many-layered learning. *Neural Computation, 14*(10), 2497–2529.
Villegas, M. A., Pedregal, D. J., & Trapero, J. R. (2018). A support vector machine for model selection in demand forecasting applications. *Computers & Industrial Engineering, 121*, 1–7. https://doi.org/10.1016/j.cie.2018.04.042.
Wu, Y., Yuan, M., Dong, S., Lin, L., & Liu, Y. (2018). Remaining useful life estimation of engineered systems using vanilla LSTM neural networks. *Neurocomputing, 275*, 167–179. https://doi.org/10.1016/j.neucom.2017.05.063.
Xin, W., Ji, W., & Chao, L. (2018). Exploring LSTM based recurrent neural network for failures time series prediction. *Journal of Beijing Unversity of Aeronautics and Astronautics, 44*(4), 772–784.
Yang, C., & Wu, Q. (2010). On stability analysis via Lyapunov exponents calculated from a time series using nonlinear mapping—a case study. *Nonlinear Dynamics, 59*(1–2), 239.
Yang, Y., Chen, H., & Jiang, T. (2015). Nonlinear response prediction of cracked rotor based on EMD. *Journal of the Franklin Institute, 352*(8), 3378–3393.
Yang, Y., Chen, Y., Wang, Y., Li, C., & Li, L. (2016). Modelling a combined method based on ANFIS and neural network improved by DE algorithm: A case study for short-term electricity demand forecasting. *Applied Soft Computing, 49*, 663–675.