An Experimental Comparison of SES, DES, TES, ARIMA, and LSTM Algorithms for Short Time Series Prediction

ภูมิพัฒน์ ดวงกลาง[1] และ ชัญญาวัจน์ สถิตภัทรสมบัติ[2]

Phummipat Daungklang[1] and Chanyawat Sathidbhattarasombad [2]

Received : January 21, 2022
Revised : March 22, 2022
Accepted : March 23, 2022

Abstract

A time series is a sequence of observations which are collected over a period: Hours, days, months, or years. There has been an assumption that the dependence of successive observations in time series is probable to exist. By means of time series analysis, this dependence is examined to discover a pattern utilised to prepare a prediction, and the prediction is one of various common objectives which can be achieved using time series analysis. Fortunately, time series analysis has been a topic of interest among academics for many decades and quite different algorithms or models for the prediction task have already been at our disposal. These algorithms are either well-established statistical ones or based on approaches of machine learning. In this paper, an experimental comparison of five different algorithms (simple exponential smoothing (SES), double exponential smoothing (DES), triple exponential smoothing (TES), autoregressive integrated moving average (ARIMA), and long short-term memory (LSTM)) was carried out using 10 examples of short time series to examine the performance of the selected algorithms over the task of

[1]ภาควิชาคอมพิวเตอร์ กองวิชาคณิตศาสตร์และคอมพิวเตอร์ กองการศึกษา โรงเรียนนายเรืออากาศนวมินทกษัตริยาธิราช
Department of Computer Science, Navaminda Kasatriyadhiraj Royal Air Force Academy
E-mail: phummipat_d@rtaf.mi.th

[2]ภาควิชาคอมพิวเตอร์ กองวิชาคณิตศาสตร์และคอมพิวเตอร์ กองการศึกษา โรงเรียนนายเรืออากาศนวมินทกษัตริยาธิราช
Department of Computer Science, Navaminda Kasatriyadhiraj Royal Air Force Academy
E-mail: chanyawat_s@rtaf.mi.th

time series prediction. The experimental results suggest that TES and seasonal ARIMA are the most appropriate to be used for time series with clear trend and seasonality. However, LSTM is the most appropriate to be used for time series without trend and seasonality.

***Keywords*:** Short Time Series Analysis, Exponential Smoothing, ARIMA, LSTM

1. Introduction

There have already been plenty of different definitions of time series. We simply state that a time series as a sequence of observations which are taken sequentially in time [1]. In our life, many sets of data appear as time series: a country's unemployment rate, the prices of a cryptocurrency, a city's electricity consumption, the number of the world's population, and so on. The observations can be made, for instance, daily, weekly, quarterly, monthly, or yearly, depending on a proposed frequency of data collection. Examples of time series exist in several different fields such as economics, finance, industry, meteorology, health, and so on. There has been assumed that a time series is affected by four main components of which a time series is supposed to consist, and which can be extracted from the observed data, namely *trend*, *cyclical*, *seasonal* and *irregular* components [2].

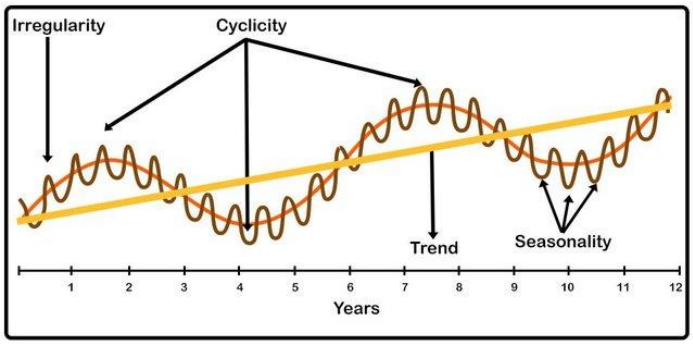

Figure 1 Illustration of Main Time Series Components [3].

A time series is called a short time series if it consists of only a few numbers of observations. Dealing with short time series can have benefits in practice because tasks or applications in the real life, in which short time series possibly exist, can be found in various disciplines, for instance, biomedicine (e.g. analysing gene expression over a short time period), economics (e.g. retail of collection at a clothing store) etc. [4, 5].

One of the purposes of "understanding time series more" is to find out from a time series whether any "pattern" is discovered which can be utilised to prepare a prediction or forecast. This can be achieved by *time series analysis*. Fortunately, time series analysis has been a topic of interest among academics for many decades and quite different algorithms for the prediction task have already been at our disposal. These algorithms are either well-established statistical ones or based on approaches of machine learning.

This paper proposes an experimental comparison of traditional statistical time series prediction algorithms and LSTM algorithm for 10 short time series each of which consists of 120 observations (100 for training and 20 for prediction evaluation). For the comparison, we will use the mean absolute percentage error (MAPE) as prediction evaluation metric.

2. Related Work

To analyse time series, many mathematical algorithms are applied. In many publications, statistical methods are also used among these methods. Besides, it is also noted that statistical methods can be exploited for short- and medium-term prediction problems. Thus, several time series prediction models were developed from statistical methods, such as autoregressive integrated moving average models (ARIMA) [7, 8], exponential smoothing algorithms [4, 9], etc.

Moreover, many new time series analysis approaches have been developed using machine learning algorithms to build a time series prediction model. Machine learning algorithms such as support vector machines (SVM) [10], and deep learning algorithms such as long short-term memories (LSTM) [11] have gained popularities with their applications in many fields and been used in many domains, and time series prediction has of course been among these domains.

3. The Methods of Research

3.1 The Data

All the time series using for our work were acquired from [12]. Each time series was made a short time series, that is, it was edited, such that only 120 observations were contained in the series. Each time series was divided into a training set and an evaluation set. The training set was used for training (fitting) the model. And the evaluation set was used for testing the model for prediction accuracy. The training set consisted of the first 100 observations, while the last 20 observations constituted the evaluation set.

3.2 The Algorithms (Models) of Interest

3.2.1 Autoregressive Integrated Moving Average Model

Two submodels are involved in ARIMA models, namely *autoregressive* (AR) models and *moving average* (MA) models. AR models use past p observation values to predict the future value of a variable which is modelled as a linear combination of these past p values. Besides, a random error at a period t (white noise) ε_t is involved in this linear combination, too [2, 13]. Mathematically, an AR model of order p, denoted by AR(p), models the current value at a period t, denoted by y_t, as

$$y_t = \varepsilon_t + \sum_{i=1}^{p} \phi_i \, y_{t-i},$$

where y_{t-1}, ..., y_{t-p} are the past p values and all ϕ_i are the weight parameters which are to be estimated from the data. The term of autoregressive indicates that it is a regression of the variable against itself [13].

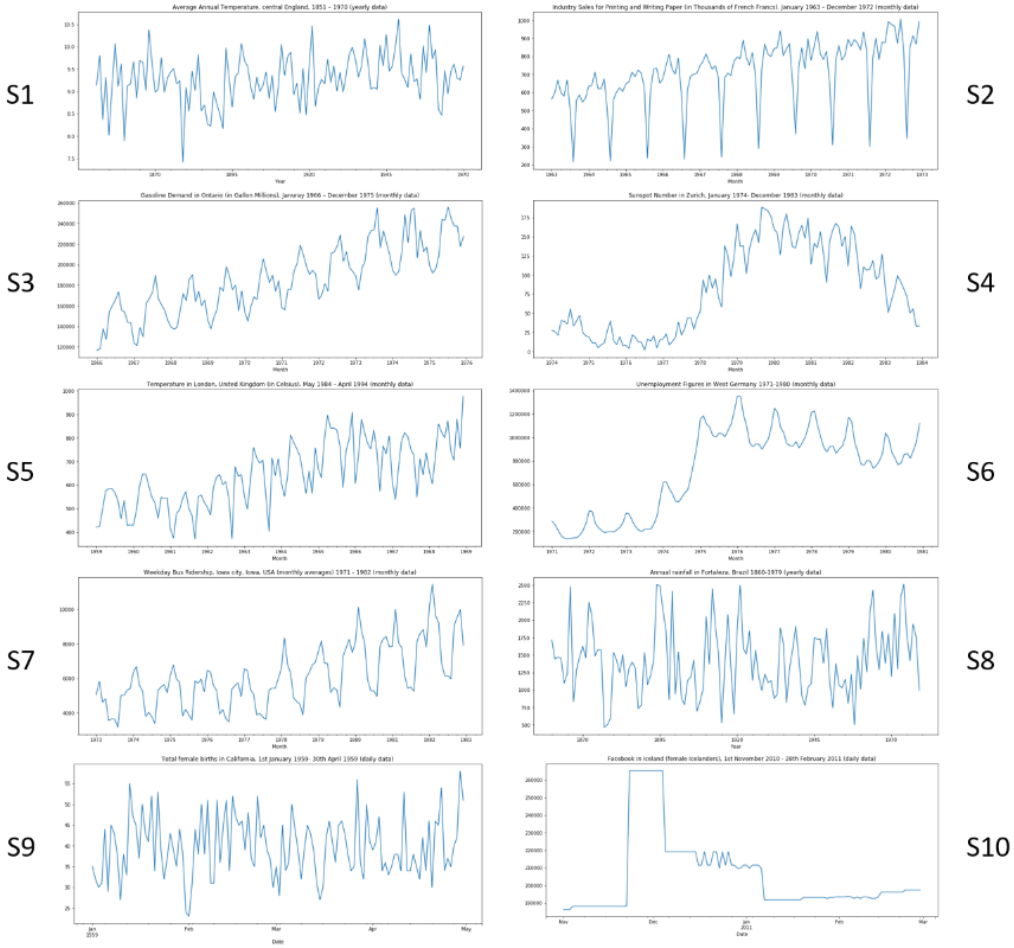

Figure 2 Plots of the Examined Time Series.

where y_{t-1}, ..., y_{t-p} are the past p values and all ϕ_i are the weight parameters which are to be estimated from the data. The term of autoregressive indicates that it is a regression of the variable against itself [13].

While AR(p) models apply past values to predict the future value of a variable, moving MA models use the past q prediction errors. An MA model of order q, denoted by MA(q) models the current value at a period t, denoted by y_t, with the white noise ε_t as

$$y_t = \varepsilon_t + \sum_{j=1}^{q} -\theta_j \, \varepsilon_{t-j},$$

where ε_{t-1}, ..., ε_{t-q} are the past q prediction errors and all $-\theta_j$ are the weight parameters which are to be estimated from the data.

Combining an AR(p) model, an MA(q) model, and the notion of *differencing* which is the progress which makes non-stationary time series stationary (we say that a time series is stationary if its probabilistic properties, such as mean, variance, autocorrelation, etc., do not change over time), by calculating the differences of consecutive observation values [13], we have an ARIMA model with differencing order d, denoted by ARIMA(p,d,q), mathematically written as

$$y_t^d = \varepsilon_t + \sum_{i=1}^{p} \phi_i \, y_{t-i}^d + \sum_{j=1}^{q} -\theta_j \varepsilon_{t-j}.$$

To determine the differencing order d, there is a most used statistical method at our disposal called the *augmented Dickey Fuller* (ADF) test which tests whether a time series is stationary or not [14]. Basically, the ADF test is a statistical significance test which involves a hypothesis testing ("reject/accept the null hypothesis") with a computed test statistic and p-value. The null hypothesis of the ADF test is that there is no stationarity in the given time series. If the p-value of the test is below the significance level α (most commonly α = 0.05), then the null hypothesis is rejected, and it can be inferred that the given time series has already been stationary, that is, d = 0. In this case, no differencing is required.

Unfortunately, the major disadvantage of ARIMA (from now on, we call it *normal* ARIMA) is that it cannot deal with time series which obviously show seasonality. Thus, there is an extension of ARIMA model to overcome this problem. A *seasonal* ARIMA model includes seasonal terms in the normal ARIMA models, mathematically denoted by ARIMA(p, d, q)(P, D, Q)m, where P, D, and Q are the autoregression, differencing, and moving average order of seasonal part. Moreover, m is the number of observations per year [13].

For instance, if a given time series presents data which are recorded quarterly, then we obtain m = 4. For this reason, one should be careful about selecting the appropriate variant for any given time series with significant seasonality.

Additionally, one must verify whether a fitted ARIMA model is a proper one by reviewing the residual error values which are produced during the model training: those values should be normally distributed, and they should not correlate with each other. But normality of the residuals is a useful, but not necessary property to be satisfied by ARIMA models [13]. The task of verifying the normality of the residual values can be achieved by the *Jarque–Bera* test [15] which is also a statistical test. In this test, the null hypothesis assumes presence of the normality. As for finding out autocorrelation in the residuals, the *Ljung–Box* test [16] is a suitable tool for it. The null hypothesis for this test assumes absence of the autocorrelation. Moreover, the number of lags to for the Ljung-Box test must be specified. A suggestion of selecting the number is min(10, T/5) for non-seasonal time series and min(2m, T/5) for seasonal time series, where T is the size of time series [17].

3.2.2 Simple Exponential Smoothing Model

Simple exponential smoothing (SES) is the simplest exponential smoothing model which is suitable to be applied for the predictions of time series which show neither trend nor seasonality [13]. Let s_t denote the *smoothed* or *fitted* value and y_t the actual value at time t. For t=1, we set $s_t = y_t$ [6]. For t = 2, ..., T, we have

$$s_t = \alpha \sum_{i=1}^{t-1} (1-\alpha)^{i-1} y_{t-i} + (1-\alpha)^{t-1} s_1,$$

where α is the *smoothing parameter* between 0 and 1 which can be thought of as weight attached to the actual observation value. The predicted future value $\hat{y}_t$ of SES models is s_t, that is, all predicted values take the last smoothed value at time t [13]. As for the value of α, we will use an optimisation tool to optimise and find the best value.

3.2.3 Double Exponential Smoothing Model

Double exponential smoothing (DES) is the extended version of SES. DES models can make predictions of time series data which consist of trends. The idea is to apply SES twice on the initial time series on the assumption that a twice application of SES to a smoothed time series may result in better prediction values. For DES models, there is another parameter, alongside the smoothing parameter α, to be calculated which is called the *trend parameter* β, a number between 0 and 1. If β is small, the trend is assumed to

change very slowly over time. If β is large, the trend shows the more rapid change [18].

In DES models, there are two equations each of which computes, like in SES models, the smoothed value s_t, and the trend b_t at time t. For t=1, we set $s_t = y_t$ and $b_t = y_2 - y_1$ [6], where y_t is again the actual value. For t = 2, ..., T, we compute s_t and b_t as in the following equations:

$$s_t = \alpha y_t + (1-\alpha)(s_{t-1} + b_{t-1}) \text{ and}$$
$$b_t = \beta(s_t - s_{t-1}) + (1-\beta)b_{t-1}.$$

Finally, the equation for the predicted value thereby can be written as

$$\hat{y}_t = s_t + hb_t.$$

Again, we will use an optimisation tool to optimise and find the best values for both α and β.

3.2.4 Triple Exponential Smoothing Model

Triple exponential smoothing (TES) is the extended version of DES. TES models have been applied to time series which exhibit not only trend, but also *seasonality*, or in other words, cyclical patterns. The idea, again, is simply that the simple exponential smoothing method is applied thrice to the initial time series to make the prediction more accurate. There are two model variations to this method the application of which depends on the nature of the seasonal pattern. That is, the *additive* model and the *multiplicative* model. The additive model is more likely to be utilised when the amplitude of the seasonal pattern appears relatively constant in time, while the multiplicative model is more likely to be preferred when the amplitude of the seasonal pattern is proportional to the level of the time series [6, 13].

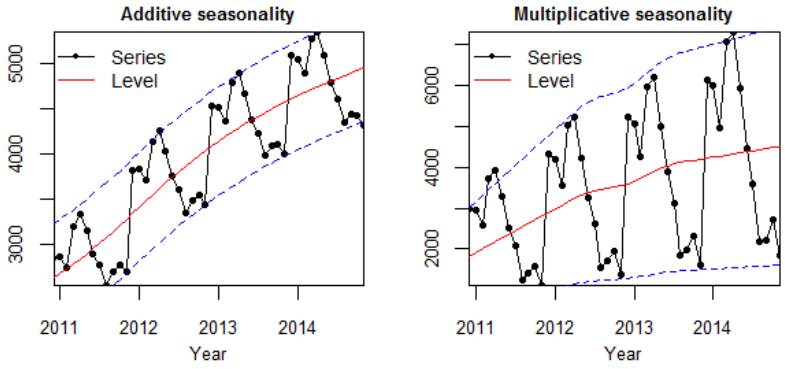

Figure 3 Additive vs. Multiplicative Seasonality [19].

Recall that we now have two exponential smoothing parameters, namely the smoothing parameter α and the trend parameter β. In TES models, a parameter for the seasonality of time series is additionally required. Hence, the seasonal parameter is defined and denoted by γ which ranges from 0 to 1. For the exact mathematical formulations, we would like to refer to such as [13] because it is quite complex, and the computations will take place using a Python program.

3.2.5 Long Short-Term Memory Network

Long short-term memory network (LSTM) is a special type of recurrent neural network (RNN). A long short-term memory network has a structure which consists of several *gated cells*. In the cells, information which is available can be stored and written. It can be read from these cells, too. The cells decide what to store, and when to allow reads, writes and erasures, by means of gates which open and close [20]. The *cell state* conveys information from past and gathering them for the present one. And information can simply go along the cell state without being changed. *Gates* are a way to optionally let information pass through or discarded. There are three gates available in a LSTM network which help control the cell state: (1) The *forget gate* decides whether the previous cell state should be forgotten, (2) the *input gate* chooses which new information needs to be stored in the cell state, and (3) the output gate generates the output which will be based on the cell state. For the exact mathematical formulations, we would like to refer to such as [19] because it is quite complex, and the computations will take place using a Python program.

3.3 Measures of Prediction Accuracy

Prediction errors play a major role in the evaluation of the models. Roughly speaking, the less the errors, the more accurate the prediction. In practice, it is important to make an appropriate choice of a measure of accuracy assessment. The mean absolute percentage error (MAPE) is recommended for its application in most publications, though it is not suitable to be used in some circumstances. For instance, MAPE is unsuitable for intermittent demand data because of the prohibited use of zero and negative actual values [21]. MAPE is defined as

$$MAPE = \frac{1}{n}\left|\sum_{t=1}^{T}\frac{y_t-\hat{y}_t}{y_t}\right|,$$

where y_t is the actual value and $\hat{y}_t$ is the predicted value at time t = 1, ..., T. We will use MAPE as the accuracy metric.

4. Results

The experiment of each model was carried out on an Intel(R) Core(TM) i7-7500 CPU processor with 2.7 GHz and a usable memory of 16 GB. Each model was written in Python. All the selected models were trained with the training set (the first 100 observations of each series) and the parameters of each model were determined using the data.

4.1 Building the Models

4.1.1 SES, DES, and TES

We used *statsmodels* [22] to build all exponential smoothing models. *statsmodels* is a Python framework which provides classes and functions for the estimation of many different statistical models, as well as for conducting statistical tests, and statistical data exploration.

For each training set, the smoothing parameter α, the trend parameter β, and the seasonal parameter γ were optimised using *hyperopt* [23]. *hyperopt* is a Python library for parameter optimising over search spaces with real-valued, discrete, and conditional dimensions. Regarding a normal optimisation procedure with *hyperopt*, a search space for each training set was defined to find the best values of the parameters by which the error during the model training was as low as possible. As stated earlier, α, β, and γ are in the range of 0 and 1. Therefore, these three parameters were searched for a real number between 0 and 1 (certainly, 0 and 1 are excluded). One of both currently implemented search algorithm of *hyperopt* which was used for our task is called *tree of Parzen estimator* (TPE). This algorithm is one of Bayesian optimisation algorithms for hyperparameters [23]. Then, the search space and the objective function returning the MAPE from the training set were passed in the minimising function to find the optimised smoothing parameter(s) which produced the least MAPE during the training. Table 1 shows the optimised values of the parameters which are rounded down to values in 8 decimal places, to be more easily presented.

Table 1. Overview of Exponential Smoothing's Parameters, ARIMA's Parameters, and LSTM's hyperparameters (e. = epochs; b.s. = batch size; n. = neurons).

ID	SES	DES		TES			ARIMA		LSTM		
	α	α	β	α	β	γ	(p,d,q)	$(P,D,Q)_m$	e.	b.s.	n.
S1	0.0626919	0.0003931	0.0000251	0.0438373	0.027798	0.0005197	(0, 0, 2)	-	100	1	2
S2	0.1030444	0.0004552	0.0001189	0.1672903	0.0003656	0.0002881	(0, 1, 1)	$(0, 1, 1)_{12}$	200	1	5
S3	0.9731978	0.9982267	0.0000353	0.0706603	0.0001303	0.0004144	(2, 1, 0)	$(1, 0, 0)_{12}$	50	3	2
S4	0.6367266	0.3175668	0.0331696	0.1927638	0.1918442	0.0004193	(0, 1, 1)	-	200	1	5
S5	0.1616712	0.2794364	0.0000388	0.7176836	0.0016582	0.0001114	(2, 1, 2)	-	200	4	3
S6	0.9999949	0.9995951	0.9992066	0.981844	0.9542871	0.0181524	(1, 1, 2)	-	100	1	2
S7	0.9986229	0.9999252	0.000008	0.4451375	0.0000836	0.0013893	(1, 1, 1)	$(0, 1, 1)_{12}$	200	5	1
S8	0.0000161	0.5098825	0.0000364	0.3080804	0.0002687	0.0000672	(1, 0, 0)	-	50	3	4
S9	0.2241311	0.0017878	0.0006171	0.0191358	0.0005383	0.0007275	(0, 0, 0)	-	100	4	5
S10	0.9999955	0.9987753	0.0023751	0.9169906	0.0010011	0.0011748	(0, 1, 0)	-	50	1	1

4.1.2 Normal and Seasonal ARIMA

In addition to the ADF test, one can use the Hyndman-Khandakar algorithm [13] to build the ARIMA models. Fortunately, there is another Python library called *pmdarima* [24] which uses *statsmodels* as its backend and hence contains several useful statistical functions including the function which can automatically build and fit the best ARIMA model, whether normal or seasonal, to a univariate time series based a provided information criterion and the necessary parameters given. With significance level α set to 0.05, we performed the ADF test on each training to see beforehand on which training set differencing was necessary, as presented in table 2.

Now, the results of the ADF test suggested that the time series S1, S8, and S9 are already stationary. Thus, the model of those three time series should be in form of ARIMA(p,0,q) Moreover, one can obviously see that there is significant seasonality in the time series S2, S3, and S7. Hence, the seasonal parameters for ARIMA were reasonably to be determined, too.

We also conducted for each fitted ARIMA model both the Jarque–Bera test and the Ljung-Box test to compute the p-value of each test, again with significant level α set to 0.05. For the latter, the number of lags was determined, as described in 3.2.1. The results of both tests are presented in table 3. We can see that all training sets passed the Ljung-Box test, that is, there is no correlation among the residuals. As for the Jarque-Bara test, nevertheless, the null hypothesis of present normality among the residuals in S3, S7,

and S10 was rejected, that is, their residuals were statistically not normally distributed. But it has been already pointed out in 3.2.1 that normality of the residuals is just a useful, but not necessary property.

Table 2. List of Each Training Set's p-Value Obtained from The ADF Test.

ID	Periodicity	p-value	Less than α?	Differencing needed?
S1	yearly	7.39×10^{-6}	Yes	No
S2	monthly	0.32	No	Yes
S3	monthly	1.0	No	Yes
S4	monthly	0.84	No	Yes
S5	monthly	0.74	No	Yes
S6	monthly	0.1	No	Yes
S7	monthly	0.98	No	Yes
S8	yearly	2.69×10^{-11}	Yes	No
S9	daily	8.87×10^{-15}	Yes	No
S10	daily	0.27	No	Yes

Table 3. Results of The Ljung-Box and The Jarque-Bara Test.

ID	Ljung-Box test				Jarque-Bara test		
	Lags	p-value (at max. lag)	Less than α?	No correlation?	p-value	Less than α?	Normality present?
S1	10	0.87	No	Yes	0.34	No	Yes
S2	20	0.66	No	Yes	0.56	No	Yes
S3	20	0.22	No	Yes	1.56×10^{-5}	Yes	No
S4	10	0.45	No	Yes	0.07	No	Yes
S5	10	0.74	No	Yes	0.19	No	Yes
S6	10	0.14	No	Yes	0.88	No	Yes
S7	20	0.67	No	Yes	1.04×10^{-17}	Yes	No
S8	10	0.73	No	Yes	0.19	No	Yes
S9	10	0.98	No	Yes	0.44	No	Yes
S10	10	0.99	No	Yes	0.0	Yes	No

4.1.3 LSTM

All LSTM models were built using mainly two Python libraries, that is, *TensorFlow* [25] and *Keras* [26]. Both are the two well established Python libraries most used for deep machine learning. Generally, to build a deep machine learning model using *TensorFlow* and *Keras*, one should consider selecting the best values for hyperparameters, such as number of *epochs*, *batch size*, and *neurons* in the hidden layer. One epoch means one forward pass and one backward pass of all the training examples and batch size is the number of training examples in one forward/backward pass. Of each time series, to find out the optimal number of epochs, batch size, and neurons, we again used *hyperopt* for this task. With the same procedure as in the case of exponential smoothing algorithms, for each time series, a search space was defined to find the best values of these hyperparameters. Due to the performance of the computer on which the experiments were performed, the number of epochs was determined between 50 and 200, the number of batch size between 1 and 5, and the number of neurons between 10 and 100. Moreover, all the numbers must be integer numbers because, for instance, there cannot 47.11 neurons exist in practice.

As for LSTM, the last 20 observations in the training set would be used for the test to avoid underfit and overfit problems. Hence, in contrast to the other training sets, the training set for LSTM would contain 80 observations. Table 1 shows the optimised hyperparameters for LSTM models of all time series.

4.2 Training and Evaluation

After the models were trained with the training set and the MAPEs were calculated, as illustrated in table 4. We see that DES was the best model for S1; TES for S2, S3, S5, S6, and S7; And LSTM for S4, S8, S9, and S10. We then used these best models to make a prediction of the last 20 observations in the evaluation set for each respective series. To see how good the best model for each training set made predictions, the results of the prediction evaluations in MAPE are provided in table 5.

Table 4. Overview of the MAPEs Calculated During Training.

ID	Mean Absolute Percentage Error (MAPE)					
	SES	DES	TES	ARIMA	LSTM	
					Training set	Test set
S1	4.61%	4.6%	5.67%	4.89%	5.31%	4.97%
S2	22.28%	20.39%	4.15%	5.65%	25.76%	26.27%

Table 4. Overview of the MAPEs Calculated During Training. (Continued)

S3	6.89%	6.9%	3.07%	3.86%	7.53%	7.14%
S4	37.89%	40.72%	46.22%	50.81%	25.8%	23.54%
S5	12.8%	12.77%	7.14%	24.02%	10.81%	11.03%
S6	9.04%	7.7%	4.75%	12.26%	6.03%	5.98%
S7	13.06%	13.15%	4.13%	5.71%	12.1%	12.65%
S8	51.73%	34.79%	31.58%	35.41%	21.77%	22.09%
S9	16.05%	62.31%	16.25%	15.9%	15.98%	15.74%
S10	1.01%	1.05%	2.46%	1.88%	0.65%	0.67%

5. Conclusions

In this paper, an experimental comparison of five models on time series prediction was carried out. The results in table 4 have revealed that TES performed best in five time series, LSTM in four time series, and DES was the best performing models in one time series. Thereafter, using the evaluation set, each model was evaluated on prediction performance which was measured as MAPE presented in table 5.

There are totally three time series which show their increasing trend and seasonality much significantly, namely time series S2, S3, and S7. Time series S5 shows its increasing trend and cyclic pattern, but both components here are not as obvious as in the three mentioned earlier. From table 4, we realise that in S2, S3, S5, S6, and S7, TES clearly outperformed the other models. In the time series with significant trend and seasonality, i.e., in S2, S3, and S7, one can clearly see that TES and seasonal ARIMA were able to achieve the similarly small MAPEs. Besides, the MAPEs which were produced by TES in the evaluation using those three time series were also considerably low.

Table 5. Overview of Prediction Performance Measurement with MAPE Using The Evaluation Sets.

ID	Best Model in Training	MAPE
S1	DES	4.63%
S2	TES	4.67%
S3	TES	3.72%
S4	LSTM	23.25%
S5	TES	7.26%

Table 5. Overview of Prediction Performance Measurement with MAPE Using The Evaluation Sets. (Continued)

S6	TES	14.42%
S7	TES	8.23%
S8	LSTM	22.17%
S9	LSTM	15.32%
S10	LSTM	0.44%

The remaining time series, namely S1, S4, S6, S8, S9, and S10, present neither clear trend nor obvious seasonality. The values of observations in each of these time series vary recognisably. Time series S4 and S6 remarkably show both increasing and decreasing trend over a period of interest. As displayed in table 4, of these six time series, LSTM outperformed the other models in four time series (S4, S8, S9, and S10). In S1, DES was surprisingly slightly better than the other models.

Consider first the time series with clear trend and seasonality. From table 4, with TES and seasonal ARIMA, each of the MAPEs was surprisingly less than 10%. In comparison, the MAPEs produced by SES, DES, and LSTM were relatively greater. It is strongly arguable that both TES and seasonal ARIMA are much more suitable models for data with clear trend and seasonality.

Consider now the time series without trend and seasonality. The MAPEs were produced by TES and DES were relatively high, which supports the suggestion over the feature of the models mentioned earlier. In S4 which shows indistinct trend and seasonality, the MAPEs produced by DES and TES were high. We argue that this can be due to that DES and TES are more likely to be used for times series with distinct trend (increasing or decreasing) and seasonality. Nevertheless, DES was the best model for S1, although the trend is visually not present in it.

To summarise, from our obtained results in table 4: TES and seasonal ARIMA are the most appropriate to be used for time series with clear trend and seasonality. LSTM is the most appropriate to be used for time series without trend and seasonality. It is quite surprising that LSTM could achieve prediction task in time series analysis well. In future work, it would be intriguing, as for LSTM, to perform such an analysis on a more powerful graphics processing unit (GPU); this would eventually increase the number of epochs, batch size, and neurons and hopefully improve the performance, too.

6. Acknowledgement

We would like to express our thanks to the journal's editors and the reviewers of this paper for their helpful and thoughtful advice some of which could certainly be used for an extension of this work in the future.

References

[1] Box GEP, Jenkins GM, Reinsel GC. Time Series Analysis: Forecasting and Control. Hoboken, New Jersey: John Wiley & Sons; 2008.

[2] Adhikari R, Agrawal RK. An Introductory Study on Time series Modeling and Forecasting [Internet]. New York: Cornell University; 2013. [updated 2013 Feb 26; cited 2022 Feb 26]. Available from: https://arxiv.org/pdf/1302.6613

[3] Pant A, Rajput RS. Time Series Analysis of Gold Price Using R. In: Digitalization in Commerce & Its Impact on Economy. [Uttar Pradesh]: Social Research Foundation; 2019. p. 15–24.

[4] Kirshners A, Borisov A. A Comparative Analysis of Short Time Series Processing Methods. Scientific Journal of RTU [Internet]. 2012 [cited 2022 Feb 26];15(1):65–69. Available from: https://itms-journals.rtu.lv/article/viewFile/v10313-012-0009-4/2677

[5] Tarsitano A. Classification Of Short Time Series [Internet]. Ponte Pietro Bucci: Universita della Calabria; 2009. [updated 2009; cited 2022 Feb 26]. Available from: http://www.ecostat.unical.it/repec/workingpapers/WP05_2009.pdf

[6] Montgomery DC, Jennings CL, Kulahci M. Introduction to Time Series Analysis and Forecasting. 2nd ed. Hoboken: John Wiley & Sons; 2015.

[7] Berninger J. Forecasting the Time Series of Apple Inc.'s Stock Price [master's thesis]. [Los Angeles]: University of California; 2018.

[8] Devi BU, Sundar D, Alli P. An Effective Time Series Analysis for Stock Trend Prediction Using ARIMA Model for Nifty Midcap-50. International Journal of Data Mining & Knowledge Management Process [Internet]. 2013 Jan [cited 2022 Feb 27];3(1):65–78. Available from: https://aircconline.com/ijdkp/V3N1/3113ijdkp06.pdf

[9] Booranawong T, Booranawong A. Simple and Double Exponential Smoothing Methods with Designed Input Data for Forecasting. Suranaree Journal of Science and Technology [Internet]. 2016 Jan-Mar [cited 2022 Feb 28];23(1):301–310. Available from: https://www.researchgate.net/publication/322355629_SIMPLE_AND_DOUBLE_EXPONENTIAL_SMOOTHING_METHODS_WITH_DESIGNED_INPUT_DATA_FOR_FORECASTING_A_SEASONAL_TIME_SERIES_IN_AN_APPLICATION_FOR_LIME_PRICES_IN_THAILAND

[10] Kim K-J. Financial time series forecasting using support vector machines. Neurocomputing [Internet]. 2003 Sep [cited 2022 Feb 28];55(1-2):307–319. Available from: https://www.sciencedirect.com/science/article/abs/pii/S0925231203003722

[11] Siami-Namini S, Tavakoli N, Namin AS. A Comparison of ARIMA and LSTM in Forecasting Time Series. In: 17th IEEE International Conference on Machine Learning and Applications. Orlando: PublisherInstitute of Electrical and Electronics Engineers; 2018. p. 1394–1401.

[12] Hyndman R, Yang Y. tsdl: Time Series Data Library v0.1.0 [Internet]. [Clayton]: Monash University; 2018 [updated 2018; cited 2022 Feb 28]. Available from: https://pkg.yangzhuo ranyang./tsdl/.

[13] Hyndman RJ, Athanasopoulos G. Forecasting: Principles and Practice. 2nd ed. Heathmont, Vic.: OTexts; 2018.

[14] Prabhakaran S. Augmented Dickey Fuller Test (ADF Test) – Must Read Guide [Internet]. [updated 2019 Nov 2; cited 2022 Feb 26]. Available from: https://www.machinelearning plus.com/time-series/augmented-dickey-fuller-test/.

[15] Jarque CM, Bera AK. Efficient tests for normality, homoscedasticity and serial independence of regression residuals. Economics Letters. 1980;6(3):255–259.

[16] Ljung GM, Box GEP. On a measure of lack of fit in time series models. Biometrika. 1978 Aug;65(2):297–303.

[17] Hyndman RJ. Thoughts on the Ljung-Box test [Internet]. [Clayton]: [Monash University]; 2014 [updated 2014 Jan 24; cited 2022 Mar 19]. Available from: https://robjhyndman.com/ hindsight/ljung-box-test/.

[18] Nau R. Moving average and exponential smoothing models [Internet]. [North Carolina]: [Duke University]; 2019 [updated 2020 Aug 18; cited 2022 Jan 13]. Available from: https://people.duke.edu/~rnau/411avg.htm#HoltLES

[19] Kourentzes N. Additive and multiplicative seasonality – can you identify them correctly? [Internet]. Lancashire: Lancaster University; 2014 [updated 2014 Nov; cited 2022 Jan 13]. Available from: https://kourentzes.com/forecasting/2014/11/09/additive-and-multiplicative-seasonality/.

[20] Goyal A. Data Science: Long Short Term Memory (LSTM) RNN [Internet]. [place unknown]: [publisher unknown]; 2018 Mar- [cited 2022 Jan 13]. Available from: https://datascience basicsblog.wordpress.com/2018/03/11/long-short-term-memorylstm-rnn/.

[21] Hyndman RJ, Koehler AB. Another look at measures of forecast accuracy. International Journal of Forecasting. 2006 Oct-Dec;22(4):679–688.

[22] statsmodels [Internet]. [place unknown]: statsmodels-developers; c2009-19 [updated 2020 Feb 08; cited 2022 Jan 13]. Available from: https://www.statsmodels.org/stable/index.html

[23] Hyperopt: Distributed Hyperparameter Optimization [Internet]. [place unknown]: GitHub, Inc.; 2022 [updated 2021 Nov 29; cited 2022 Jan 13]. Available from: https://github.com/ hyperopt/hyperopt

[24] pmdarima: ARIMA estimators for Python [Internet]. [place unknown]: Taylor G Smith; c2017-22 [updated 2017 Sep 3; cited 2022 Feb 26]. Available from: https://alkaline-ml.com/ pmdarima/index.html

[25] TensorFlow [Internet]. [place unknown]: GitHub, Inc.; 2022 [updated 2021 Nov 29; cited 2022 Jan 13]. Available from: https://www.tensorflow.org/.

[26] Keras [Internet]. [place unknown]: [publisher unknown]; 2022 [updated 2021 Nov 29; cited 2022 Jan 13]. Available from: https://keras.io/.