ORIGINAL ARTICLE

Enhancing LSTM performance for financial time series forecasting: the differencing approach - Diff-LSTM

Levent Gul · Ebru Yuksel Haliloglu · Hande Sevgi · Nukhet Dogan · M. Hakan Berument

Received: 20 November 2025 / Accepted: 12 July 2026
© The Author(s) 2026

Abstract
Long Short-Term Memory (LSTM) networks are increasingly preferred over conventional econometric models for forecasting non-linear time series such as stock market data. This study proposes the "Diff-LSTM" approach, designed to enhance the forecasting capabilities of standard LSTM networks for non-stationary time series such as stock market data. Unlike conventional applications, the Diff-LSTM methodology trains the network on first-differenced – hence, stationary- series from which forecasts for the original series (in levels) are reconstructed by updating with the observed previous value of the level series. In LSTM networks, due to the limitations of activation functions such as *sigmoid* and *tanh*, a downward bias can be observed in forecasts of non-stationary series that have a unit root. Therefore, the proposed Diff-LSTM approach mitigates this downward bias, enabling more reliable forecasts. Diff-LSTM models are evaluated for forecasting accuracy across five major U.S. stock market indices using three cross-validation schemes and four performance measures. The current study demonstrates the superiority of the Diff-LSTM model, which outperformed standard LSTM models in 56 of 60 evaluated instances. This suggests that using Diff-LSTM models improves the forecasting performance considerably.

Keywords Cross-validation · Long Short-Term Memory · Machine Learning

JEL Classification C22 · C45 · G17

1 Introduction

Recent advances in technology have significantly enhanced the capability to systematically store and analyze large amounts of time-series data. Although traditional econometric models—such as Autoregressive Integrated Moving Average (ARIMA), Vector Autoregression (VAR), and Generalized Autoregressive Conditional Heteroskedasticity (GARCH)—have proven effective for financial time series, they often struggle with complex, non-stationary and non-linear data structures [1, 12]. Consequently, there is a growing need for models capable of adapting to these complex, non-stationary and non-linear characteristics. In this context, the Long Short-Term Memory (LSTM) networks, with their short- and long-term memory capabilities, very suitable for time series, have become a very popular deep learning alternative to traditional econometric models [23, 53]. Developed by

This paper is partially based on the PhD thesis of Levent Gul at Ankara Hacı Bayram Veli University.



Published online: 11 August 2026

Hochreiter and Schmidhuber [23], the LSTM model is a recurrent neural network architecture explicitly designed to capture long-term dependencies in time-series data. This allows it to propagate weights across layers, maintain the stability of the gradient, and learn continuously. LSTM is a common method for predicting financial data like stock prices, cryptocurrencies, commodity prices and exchange rates (see, for example, Kohzadi, [31, 22, 24, 26, 29, 39, 40, 47, 55, 56, 57, 59, 62]). Comprehensive surveys on LSTM and its applications indicate that LSTM has achieved widespread adoption across various domains (see, e.g. Kim and Nam, [30], Yu, [63, 21]).

This study introduces the Diff-LSTM approach to improve LSTM forecasting performance by addressing the output-range limitation of LSTM cell activation functions. A typical LSTM unit consists of *sigmoid* and *hyperbolic tangent* (*tanh*) activation functions, which generate outputs within the ranges of (0,1) and (-1,1), respectively. Mathematically, the outputs of these two activation functions asymptotically approach their threshold values but never reach these threshold values. This causes two problems: one is systematic bias towards lower predictions, and the second is the saturation of functions and their reduced sensitivity to large inputs. The adapted Diff-LSTM approach addresses these limitations by modeling the logarithmic difference and adding the previous observation's value, rather than modeling the logarithm of the series. Therefore, Diff-LSTM models the return predictions, thereby more effectively leveraging the model's capacity to capture the complex short- and long-term patterns, non-linearity, and volatility characteristic of financial time series data. The main motivation for the proposed Diff-LSTM approach is conceptually rooted in the seminal work of Dickey [13] and Dickey and Fuller [14] on unit root tests. In their unit root test, the persistent downward bias in the autoregressive parameter, consistently found to be less than 1, was addressed. They proposed differencing the time series as a solution. Similarly, our proposed Diff-LSTM suggests differencing the non-stationary input data before feeding it to the LSTM network to eliminate the downward prediction bias and the sensitivity to large data points inherent in LSTM cells. This paper demonstrates that the Diff-LSTM approach achieves superior forecasting performance compared to a standard LSTM network.

We conducted extensive empirical comparisons between our proposed Diff-LSTM approach and conventional LSTM networks to verify whether our model enhances LSTM forecasting performance. Five major US stock market indices -the NASDAQ Composite, NYSE Composite, Russell 2000, S&P 500, and Wilshire 5000- were employed to test the performance of the proposed approach with various financial market conditions. The models' performance was evaluated using four forecast accuracy measures: mean absolute error (MAE), mean absolute percentage error (MAPE), mean squared error (MSE), and root mean squared error (RMSE). Our results are also robust with the Diebold–Mariano test statistic. An important aspect of our work was the application of three distinct cross-validation methods to ensure a thorough evaluation of model reliability. The first cross-validation technique was the traditional time-series splitting (TS), which splits the first 90% of observations into the training set and the last 10% into the test set. The second validation method used was K-fold time series splitting (FTS) with 10 folds. In this case, the data set was divided into 10 equal parts. The data was trained to a certain point and then tested on the next segment in each iteration - thus preserving the chronological order. The last cross-validation method was blocked *K*-fold time-series splitting (BFTS) with 5 folds, where each fold was used as a test set and the others for training. Both the training and test sets retain the time series data order to prevent information leakage across folds.

The empirical evidence on the standard LSTM and the proposed Diff-LSTM model provided here suggests that first, the Diff-LSTM provides a significant performance improvement, outperforming the standard LSTM in 56 of the 60 total performance metrics calculated, which constitutes a superiority in approximately 93% of all comparisons (the only exception was its marginally lower performance on four metrics specific to the NYSE Composite index under the TS cross-validation). Second, regarding model stability: standard LSTM performance tends to degrade across datasets, whereas the Diff-LSTM approach reports robustness statistics. This also suggests that the Diff-LSTM model is less sensitive to changes in data structure and can perform well even on smaller data partitions.

The rest of the paper is organized as follows: Sect. 2 introduces the LSTM model. In Sect. 3, the study's motivation and the proposed forecasting approach are explained. In Sect. 4, the dataset is introduced, and the stationarity of the time series is investigated. In Sect. 5, cross-validation and model selection criteria are given. Empirical evidence is presented in Sect. 6. Extensions of the basic specifications are explored in Sect. 7. The results are summarized in Sect. 8.

2 The LSTM model

Standard machine learning algorithms (such as Restricted Boltzmann Machine, Deep Convolutional Neural Networks, Convolutional Neural Network, Deep Belief Network, Stacked Auto-Encoder, and Recurrent Neural Network) work successfully in many important application areas such as image and speech recognition, natural language processing, and big data studies (see, for example, [8, 11, 32, 64]). However, these models do not apply to time-series analysis because of the autoregressive dependencies between sequential observations. Therefore, Recurrent Neural Networks (RNNs) have been developed to forecast time series data by incorporating memory mechanisms [48, 50, 52]. RNNs face the problems of exploding gradients or vanishing gradients due to their recursive structure and deep layers.

Figure 1 illustrates the structure of an RNN. In this figure, x_t is the input, h_t is the output, S_t is the hidden layer, v_x is the weight of the input, v_h is the weight of the output and v_r is the iteration weight. The hidden state S_t of each step propagates forward with the time series, and S_t is a function of v_r and v_x, so there will be continuous calculation of the indirect hidden state between each time step [16]. The case $v_r > 1$ causes an exploding gradient, while the case $v_r < 1$ causes a vanishing gradient. In RNN training, exploding gradients occur when the output values become too large, and vanishing gradients occur when the output values approach 0. During RNN training, exploding gradients can lead to excessively large weight updates and thus, the network might not converge to an optimal solution. Conversely, the vanishing gradients may cause exceedingly slow or entirely stagnant weight updates [23]. To address these problems, the LSTM method was introduced by Hochreiter and Schmidhuber [23] and has been widely used for time series forecasting [61]. LSTM is a type of RNN that stores information about past observations for future use (Patterson, [45]). LSTM networks have memory cells, input gates, output gates and forget gates to store long term information and control the flow of information between memory cells to store and retrieve the required information as needed [20]. Thus, it manages information storage and prevents exploding/vanishing gradient problems as in RNNs. LSTM solves the long-term dependency problem and can be considered as a better RNN architecture with a feedback loop. Thus, it can process not only individual data points but also entire data series [23, 37]

A visual representation of the LSTM mechanism is given in Fig. 2. The LSTM architecture given above consists of a memory cell and three gates. In Fig. 2, $+$ and $\times$ denote addition and multiplication in the model, respectively. σ stands for the *sigmoid* activation function, *tanh* denotes the *tanh* activation function, and the black

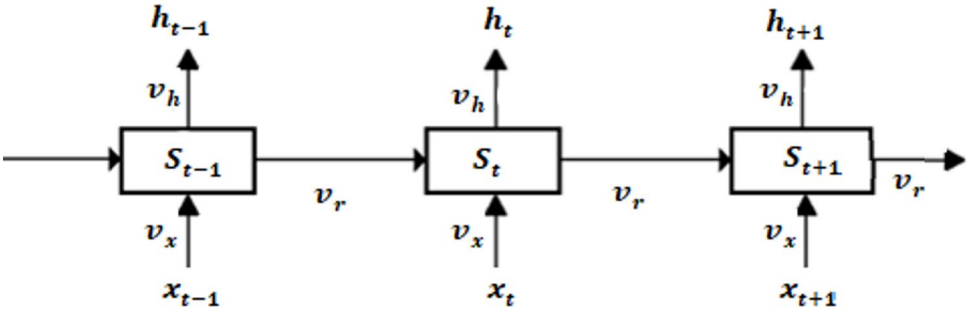

Fig. 1 The Structure of RNN

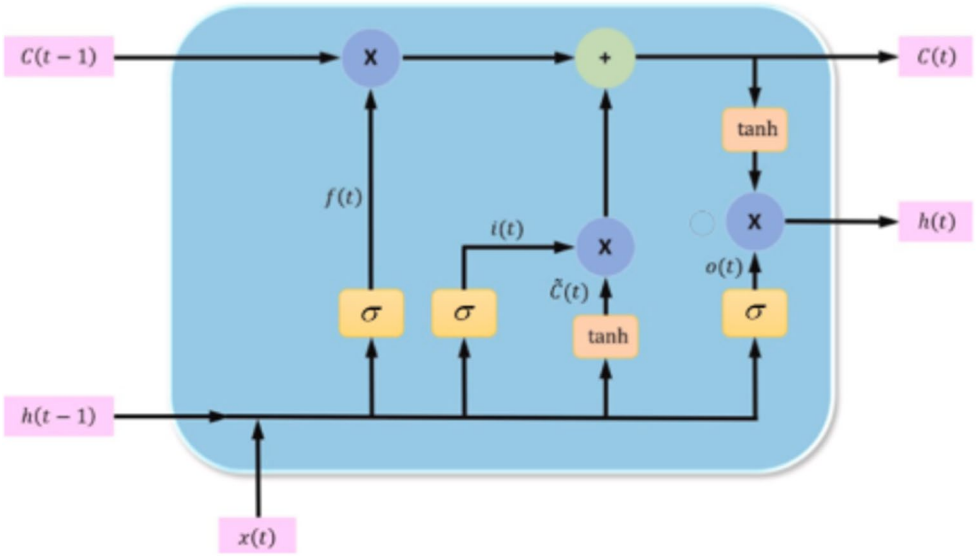

Fig. 2 The LSTM Mechanism. Source: [41].

arrows show the direction of information flow. The equations of the *sigmoid* (Eq. 1) and *tanh* (Eq. 2) activation functions are as follows:

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

$$tan\ h\,(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{2}$$

where x represents the value to be transformed.

Activation functions modify input signals into representations that improve model learning and performance. The *sigmoid* function transforms an input value into a value between 0 and 1, with 1 meaning the network leaves the input unchanged and 0 meaning it totally forgets it. The *tanh* function operates similarly to the *sigmoid* except that the possible outputs range from − 1 to 1 instead of from 0 to 1 [15].

The LSTM structure consists of three gates, each with a different task. The forget gate generates an output ranging from 0 to 1, where 1 indicates "completely retain new information," and 0 indicates "completely forget new information." The mathematical definition of the forget gate is as follows:

$$f_t = \sigma\ (W_f x_t + U_f h_{t-1} + b_f) \tag{3}$$

where f_t denotes the forgetting threshold at time t, W_f and U_f denote the weights, x_t denotes the input value, h_{t-1} denotes the output value at time *t-1*, and b_f is the bias term of the forget gate.

The memory gate chooses which new data point to store in a cell. The mathematical definition of the memory gate is given by

$$i_t = \sigma\ (W_i x_t + U_i h_{t-1} + b_i) \tag{4}$$

$$\widehat{C}_t = tanh\,(W_c x_t + U_c h_{t-1} + b_c) \tag{5}$$

$$C_t = f_t C_{t-1} + i_t \widehat{C}_t \tag{6}$$

where i_t denotes the *sigmoid* layer, $\widehat{C}_t$ denotes the *tanh* layer, and C_t denotes the state of the cell at time *t*. b_i is the bias term of the input gate and b_c is the bias term of the *tanh* layer. In the memory gate, first, the "input door layer" chooses the input values to be modified through a *sigmoid* layer. Next, a vector of new candidate values that could be added to the state is computed by a *tanh* layer.

The output gate determines what is yielded from each cell.[1] The output value is computed using both the cell state and the filtered new data point. The mathematical expression of the output gate is denoted by

$$O_t = \sigma\ (W_o x_t + U_o h_{t-1} + b_o) \tag{7}$$

where b_o is the bias term of the output gate.

Lastly, the output value of the memory cell is shown by

$$h_t = O_t tanh\,(C_t)\,, \tag{8}$$

where h_t is the output value of the cell at time *t*.

Through their integrated gate mechanisms, LSTM networks mitigate the gradient problems that plague RNNs, thereby enabling the accurate processing of long-term temporal dependencies. Furthermore, these gate structures enable the training of deeper network architectures compared to conventional RNNs.

3 Motivation and proposed forecasting approach

In a classical linear regression model, when the Gauss–Markov assumptions hold, the OLS estimator of the coefficients is the Best Linear Unbiased Estimator (BLUE). However, as a special case of classical linear regression, an autoregressive (AR) process does not meet the BLUE condition because lagged values of the dependent variable are also used as predictors [43]. Bias in estimating AR parameters has been extensively studied [6, 38, 42, 49]. The findings of this line of research indicate that AR parameters are biased by serial correlation, and the bias increases as the true value of the AR parameter approaches 1. Also, the AR parameter is consistently underestimated. Orcutt and Irwin, [42] first explicitly identified OLS bias in the estimation of the AR parameter. They showed that the bias arises from serial correlation between the lagged values of the dependent variable originates from two sources. Initially, if the actual mean of the series is known, the serial correlations will generally be biased unless the related autocorrelations are 0. Since the serial correlation and the estimated variance are not independent in general, the bias appears to be towards 0. In fact, the mean is often calculated from the sample, which introduces another bias. This secondary source of bias, which persists even with 0 autocorrelation, is consistently negative in an extensive series; specifically, the expectation of the serial correlation computed from the estimated sample mean is invariably algebraically lower than that computed from the known true mean, and further it gets closer to -1. Similarly, Sastry , [49] studied the biasness in OLS estimation of AR parameters. However, they studied only the second case mentioned in Orcutt and Irwin [42], and the results were similar. Marriott and Pope, [38], Hurwicz [25], Kendall [28], White [58], Pantula and Fuller [44], Yamamoto and Kunitomo [60], Shaman and Stine [51], Le Breton and Pham [33], Sims and Uhlig [54], and Andrews [3] further elaborated this well-documented problem in the literature.

This problem is more vital for testing a unit root. To be specific, consider a first-order AR process, AR(1), given as follows:

$$x_t = \rho\ x_{t-1} + \epsilon_t \tag{9}$$

in which ϵ_t are normally distributed random variables and ρ is the unit root parameter. Dickey, [13] and Dickey and Fuller [14] recognized this problem and proposed estimating

[1] Note that f_t, i_t and $\widehat{C}_t$ will always be less than 1.

$$\Delta\ x_t = \theta_1 x_{t-1} + \epsilon_t \tag{10}$$

where $\Delta\ x_t = x_t - x_{t-1}$ and $\theta_1 = (\rho - 1)$ for their unit root tests.

The non-stationarity of the series is tested using unit root tests. If $\rho = 1$, then the series is non-stationary, while if $\rho < 1$, the series does not have a unit root. However, Dickey , [13] also showed in his Monte Carlo study that OLS estimates are biased downward when the unit root parameter in Eq. (9) is both stationary ($\rho < 1$) and non-stationary ($\rho = 1$). Moreover, Fuller [19] and Dickey [13] showed that the probability of $\widehat{\rho} < 1$ when $\rho = 1$ is 0.6826. Thus, there may be situations in which non-stationary series are incorrectly treated as stationary. Further, Dickey [13] and Dickey and Fuller [14] suggested that if there is a constant term, as well, in Eq. (9) with $\rho = 1$, the ordinary least squares estimator of the constant term will not be distributed normally, and there is a correlation between the constant term and the unit root parameter.

To avoid these problems, the model in Eq. (10), which uses first-differenced series rather than level values for the left-hand side of the specification, is recommended for unit root testing. Thus, Eq. (10) can be employed to determine whether the series has a unit root by the parameter θ_1 being equal to 0. If $\theta_1 = 0$, the series has a unit root, while the opposite, if $\theta_1 < 0$. Using Eq. (10), not only is the bias in OLS estimation eliminated, but also two independent estimators with easily interpretable null distributions are obtained.

Similar to the misleading bias in unit root tests, one can observe a downward bias in LSTM cells, because of the inherent limitations of the *sigmoid* and *tanh* activation functions. A standard LSTM architecture uses three *sigmoid* and two *tanh* activation functions. The *sigmoid* function has an output range of (0, 1) while the tanh function has an output range of (-1, 1). The *tanh* function can asymptotically approach the limiting values − 1 and 1, but it cannot take these values exactly. The *sigmoid* function has an upper bound of 1 which is asymptotically reached slower than *tanh*, resulting in a limited output range. All these features of the *sigmoid* and *tanh* functions in the LSTM cell make the activation functions sensitive to small input values, potentially diminishing predictive accuracy for high-frequency data, such as stock prices. The output patterns of the *sigmoid* and *tanh* functions across varying input values are given in detail in Table 1.

Table 1 Output Values and Changes of *tanh* and *sigmoid* Functions According to Various Input Values

Input	Output	Change	Input	Output	Change	Input	Output	Change
sigmoid								
0.1	0.5249791875	0.0249791875	**1.1**	0.7502601056	0.0192015270	**10.1**	0.9999589221	0.0000043200
0.2	0.5498339973	0.0248548098	**1.2**	0.7685247835	0.0182646779	**10.2**	0.9999628311	0.0000039089
0.3	0.5744425168	0.0246085195	**1.3**	0.7858349830	0.0173101995	**10.3**	0.9999663680	0.0000035370
0.4	0.5986876601	0.0242451433	**1.4**	0.8021838886	0.0163489055	**10.4**	0.9999695684	0.0000032004
0.5	0.6224593312	0.0237716711	**1.5**	0.8175744762	0.0153905876	**10.5**	0.9999724643	0.0000028959
0.6	0.6456563062	0.0231969750	**1.6**	0.8320183851	0.0144439089	**10.6**	0.9999750846	0.0000026203
0.7	0.6681877722	0.0225314659	**1.7**	0.8455347349	0.0135163498	**10.7**	0.9999774556	0.0000023710
0.8	0.6899744811	0.0217867090	**1.8**	0.8581489351	0.0126142002	**10.8**	0.9999796009	0.0000021453
0.9	0.7109495026	0.0209750215	**1.9**	0.8698915256	0.0117425905	**10.9**	0.9999815421	0.0000019412
1	0.7310585786	0.0201090760	**2**	0.8807970780	0.0109055523	**11**	0.9999832986	0.0000017565
tanh								
0.1	0.0996679946	0.0996679946	**1.1**	0.8004990218	0.0389048658	**10.1**	0.9999999966	0.0000000007
0.2	0.1973753202	0.0977073256	**1.2**	0.8336546070	0.0331555853	**10.2**	0.9999999972	0.0000000006
0.3	0.2913126125	0.0939372922	**1.3**	0.8617231593	0.0280685523	**10.3**	0.9999999977	0.0000000005
0.4	0.3799489623	0.0886363498	**1.4**	0.8853516482	0.0236284889	**10.4**	0.9999999981	0.0000000004
0.5	0.4621171573	0.0821681950	**1.5**	0.9051482536	0.0197966054	**10.5**	0.9999999985	0.0000000003
0.6	0.5370495670	0.0749324097	**1.6**	0.9216685544	0.0165203008	**10.6**	0.9999999988	0.0000000003
0.7	0.6043677771	0.0673182101	**1.7**	0.9354090706	0.0137405162	**10.7**	0.9999999990	0.0000000002
0.8	0.6640367703	0.0596689932	**1.8**	0.9468060128	0.0113969422	**10.8**	0.9999999992	0.0000000002
0.9	0.7162978702	0.0522610999	**1.9**	0.9562374581	0.0094314453	**10.9**	0.9999999993	0.0000000002
1	0.7615941560	0.0452962858	**2**	0.9640275801	0.0077901219	**11**	0.9999999994	0.0000000001

Table 1 presents the observed variations in the *sigmoid* and *tanh* functions across multiple input values. It is seen that no matter how high is the input value, the output of both functions cannot exceed their upper limits imposing downward biases in the estimations. Furthermore, for both functions, changes are negligible after specific threshold points. The results also showed that the *tanh* function converges to its upper limit faster than the *sigmoid* function.

To address the aforementioned limitations of the *tanh* and *sigmoid* activation functions, resolve biasness issue and improve the predictive accuracy of LSTM models, the Diff-LSTM approach is proposed, motivated by the methodologies of Dickey [13] and Dickey and Fuller and [14]. The Diff-LSTM approach argues that using the logarithmic differences of time series (Δx_t), rather than their log-level values (x_t), can improve LSTM-based forecasting accuracy. This will lead to larger magnitudes of changes inside the proposed Diff-LSTM framework and thus, the activation functions can better capture the changes in input. According to the Diff-LSTM approach, the redefined version of the LSTM model based on Δx_t is as follows:

$$f_t = \sigma\ (W_f \Delta x_t + U_f h_{t-1} + b_f) \tag{11}$$

$$i_t = \sigma\ (W_i \Delta x_t + U_i h_{t-1} + b_i) \tag{12}$$

$$\widehat{C}_t = \ tanh\,(W_c \Delta x_t + U_c h_{t-1} + b_c) \tag{13}$$

$$C_t = f_t C_{t-1} + i_t \widehat{C}_t \tag{14}$$

$$O_t = \sigma\ (W_o \Delta x_t + U_o h_{t-1} + b_o) \tag{15}$$

$$h_t = O_t tanh\,(C_t) \tag{16}$$

The following Eq. (17) presents the computation of forecasted values within the proposed Diff-LSTM framework:

$$\widehat{G}_t = \widehat{\Delta x_t} + x_{t-1} \tag{17}$$

where $\widehat{G}_t$ is the forecasted value of the proposed Diff-LSTM approach, $\widehat{\Delta x_t}$ is the LSTM forecasted value, and x_{t-1} is the one-period lag of the inputs. In Eq. (17), true observed values are fed to the model as the previous step (x_{t-1}) input in order to gather one-step forecasting.

In many financial applications, practitioners either work with log levels without explicitly addressing the possible unit-root behavior of the series or apply log differencing as a generic preprocessing step to induce approximate stationarity, after which a standard LSTM is trained on the transformed series. By contrast, the proposed Diff-LSTM is formulated as an explicit difference-based forecasting rule: the network is trained on the log-differenced series, and then one-step-ahead level forecasts are reconstructed via $\widehat{G}_t = \widehat{\Delta x_t} + x_{t-1}$, where $\widehat{\Delta x_t}$ denotes the LSTM forecast of the log-difference and x_{t-1} is the previous observed log-level. This addresses not only standard non-stationarity concerns but also the bounded and saturating nature of the *sigmoid* and *tanh* activation functions, which tend to induce downward bias when the LSTM is trained directly on large, trending level series. Working with log differences as a first step reduces the effective variation in the inputs to these activation functions, alleviating saturation, while the reconstruction step ensures that model evaluation and comparison are always conducted at the level domain relevant to practitioners. Hence, Diff-LSTM should be understood as a structured differencing-and-reconstruction architecture rather than a conventional LSTM applied to log levels or a log-differenced input series.

The incorporation of $\widehat{\Delta x_t}$ within the Diff-LSTM framework, it eliminates the initial data preparation and transformation procedures, which are mostly time-consuming and computationally expensive. In general, data transformations are used to improve forecasting accuracy, ease model training, reduce the effect of outliers, reduce data dimensionality and improve data distribution. Moreover, these transformations are often combined

with the selection of methodologies that are adapted to the characteristics of the dataset and the intended purpose of the model. However, the proposed forecasting approach shows that the model's performance could be improved by applying the differencing operation without any pre-processing or dataset transformation. Furthermore, while existing research attempts to improve LSTM performance by incorporating returns or change rates rather than (logarithmic) levels (see, for example, [18, 34]), Diff-LSTM offers a novel, and in particular more practical, methodology.

4 Data

The stock market indices used to compare forecasting performance are the S&P 500, NYSE Composite, Wilshire 5000, NASDAQ Composite, and Russell 2000, which are the five leading US indices and are frequently used in such studies. The indices are gathered from Yahoo Finance. These indices consist of companies at different sizes, various industries and have diverse volatility patterns. The S&P 500 Index is based on the market capitalization of the largest 500 publicly traded companies in the United States. The Russell 2000 Index is based on the market capitalization of the smallest 2000 publicly traded companies in the Russell Index. The Wilshire 5000 is calculated based on the market value of all actively traded US stocks, and the number of stocks it tracks varies depending on the stocks traded. The NASDAQ Composite Index is a market-cap-weighted index of more than 3700 stocks traded on the NASDAQ exchange, mostly in the technology sector. Finally, the NYSE Composite Index is calculated by including the stocks traded on the New York Stock Exchange.

Daily closing values of the indices were used in the analyses. The starting dates of the indices are as follows: S&P 500, 3 January 1928; NYSE Composite, 3 January 1966; Wilshire 5000, 4 January 1989; NASDAQ Composite, 8 February 1971; and Russell 2000, 11 September 1987. While the start dates of the index data vary, the end date of all indices is 3 September 2024.

Given the output range limitations of the *sigmoid* and *tanh* functions, which are (0,1) and (-1,1), respectively, it is essential to assess the unit-root characteristics of the variables under consideration. For this reason, the stationarity of the five indices used in the comparisons is investigated with the Augmented Dickey–Fuller (ADF) and Phillips–Perron (PP) unit root tests and the Kwiatkowski–Phillips–Schmidt–Shin (KPSS) stationarity test. The test statistics are provided in Table 2.

The null hypotheses of the ADF and PP unit root tests are the presence of a unit root in the series, while the null hypothesis of the KPSS stationarity test is the stationarity of the series. In Table 2, the ADF and PP unit root tests indicate that all the series have a unit root at levels. On the other hand, the series after first differencing does not have the unit root at 1% level of significance. Likewise, the null hypothesis of the KPSS test is rejected for the level series at the 1% significance level, while it cannot be rejected for the differenced series. The test results for all series therefore suggest that the series are integrated of order 1, I(1).

Table 2 Unit Root/Stationarity Test Statistics

		ADF	PP	KPSS
		Test Statistic	Test Statistic	Test Statistic
NYSE Composite	**Level**	-0.192	-0.188	19.790*
	First Difference	-20.544*	-119.146*	0.071
S&P 500	**Level**	0.730	0.733	25.327*
	First Difference	-22.459*	-157.680*	0.216
NASDAQ Composite	**Level**	0.044	0.028	18.098*
	First Difference	-17.834*	-114.603*	0.073
Russell 2000	**Level**	-1.367	-0.774	15.257*
	First Difference	-17.760*	-99.246*	0.018
Wilshire 5000	**Level**	-0.570	-0.673	13.832*
	First Difference	-17.143*	-100.713*	0.073

* indicates that the test statistics exceed the critical value at the 1% significance level. All test specifications include constant terms

5 Modeling framework

In the present study, comparative analyses using three distinct cross-validation methodologies are conducted to compare the forecasting performance of the proposed Diff-LSTM approach relative to that of the LSTM. In the following subsections, the definition of performance criteria, the details of the cross-validation methods and the hyperparameter optimization process for the LSTM models are explained.

5.1 Performance criteria

In this study, performance criteria serve a dual purpose: evaluating the LSTM model's individual performance and identifying the superior model during comparative analyses. While RMSE is utilized to measure standalone model performance, for model comparison a broader set of metrics—RMSE, MSE, MAE, and MAPE—is employed. Although a single metric suffices for assessing an individual model's accuracy, using multiple metrics provides a more robust basis for comparison. Consequently, error-based criteria are prioritized, and four distinct metrics with varying sensitivities are applied.

$$MAE = \frac{1}{n}\sum_{t=1}^{n} |e_t| \tag{18}$$

$$MSE = \frac{1}{n}\sum_{t=1}^{n} e_t^2 \tag{19}$$

$$RMSE = \sqrt{MSE} \tag{20}$$

$$MAPE = \frac{1}{n}\sum_{t=1}^{n} \frac{|e_t|}{|x_t|} \tag{21}$$

where e_t refers to forecast error, x_t is observed value of the dependent variable, and n is the number of observations in the forecasting period.

The performance metrics are applied in their standard forms, as presented in the equations, without any restrictions. As for MAPE, no clipping or smoothing is applied even for very small denominators. The metrics are calculated on the reconstructed level series. Each performance criterion employed presents distinct advantages and limitations in practice. While MAE is often preferred among the four metrics for its ease of calculation and interpretation, MSE is notably more sensitive to outliers within the error terms [5, 10, 36]. On the other hand, although RMSE and MSE generally lead to similar results, RMSE is generally more sensitive to outliers than MSE [4]. The dimensionless MAPE metric can take infinite or extremely large values if error terms are 0 or tend to 0. A further disadvantage of MAPE is that it penalizes outlier estimates more severely than those closer to the sample mean [17].

5.2 Cross-validation

Cross-validation is a methodology for assessing model performance by partitioning the data into distinct subsets. Recently, it has been widely used in machine learning for hyperparameter tuning, data partitioning, and model comparison. Primarily, cross-validation enhances model accuracy and facilitates selecting the best-performing model. In the present study, cross-validation methods are used to select the model that exhibits the best performance in comparisons. For this, the TS, FTS, and BFTS methods, which are frequently used for time series, are used in our analyses. It should be emphasized that, in literature, mostly one and rarely two of these methods are used together for model justifications. However, herein, all three are used together to provide a more comprehensive evaluation of the proposed approach's performance.

The TS method is commonly employed as a standard dataset partitioning strategy in machine learning models applied to time series data. To address the autoregressive dependency in time series data, the TS method divides the dataset into training and test sets. Generally, the initial larger partition is devoted to training the model, and the remaining smaller partition is used for testing. This approach acknowledges the inherent sequential nature of time series data.

In the FTS method, the current learning set is divided into K disjoint subsets of approximately equal sizes. Here, "folding" refers to the number of resulting subsets. This division is performed by considering the temporal dependency among the data without replacement. The model is applied to the *K-1* subsets, which together represent the training set. Then, the model is applied to the remaining subset, the validation set, and performance is measured. This procedure is repeated until each of the K subsets serves as the validation set. The average of the K performance measurements on the K validation sets is the cross-validated performance.

The FTS procedure can be summarized as follows:

1. Split the dataset into K groups, considering the temporal dependency.
2. For each unique group:

    a. Firstly, the training set and test set for the first fold are determined.
    b. In the following folds, the test set of the previous fold is added to the training set, and the time series continues without interruption until the last fold.
    c. Fit a model on the training set and evaluate it on the test set.
    d. Retain the evaluation score and discard the model.

3. Summarize the performance of the model using the sample model evaluation scores.

The question we address here is how to choose a reasonable value of K that achieves a practical balance between bias, variance, and computation time, so that cross-validation yields a reliable assessment of models' forecasting performance. These issues have been largely ignored in the literature, particularly with discrete Bayesian networks. Instead, 10-fold ($K = 10$) is commonly used without any particular test or specific rationale (e.g., [2, 7, 65]). Breiman and Spector, [9] used expected squared error as the classification error for simulated analytic models they tested and found that the sub-model selection criteria greatly affected validation results, with 5-fold cross-validation providing better results. Therefore, it is important to use an optimal K value to meet the low bias, low variance, and high accuracy requirements of the forecasting models.

Third, the BFTS method is applied similar to the FTS method. However, in the BFTS method, the time series is not continued uninterrupted across folds to prevent back leakage. The BFTS procedure divides the dataset into K non-overlapping folds, accounting for temporal dependencies. A training set and a test set are determined for each fold, and K separate models are trained according to the folds. As a result, the forecasting performance criterion for each fold is calculated separately, and the average across folds is used for model evaluation. Non-overlapping folds break the connections between them and prevent back-leakage of information.

For the present study, within the TS cross-validation framework, the initial 90% of the dataset was allocated to the training set and the remaining 10% to the test set. In the FTS and BFTS cross-validations, the K values are selected as 10 and 5, respectively.

5.3 Parameter optimization for LSTM

Parameter optimization for LSTM models is an important part of effective model training. Optimal parameters have been demonstrated to significantly enhance the learning speed, prediction accuracy, and generalization capabilities of machine learning models, while suboptimal parameters can severely impair their efficiency. Various methods are used for parameter optimization, including Bayesian optimization, genetic algorithm, grid search, manual tuning and random search. In addition, Keras-Tuner hyperparameter configurations, merging the

strengths of random search and Bayesian optimization, has emerged as a widely adopted hyperparameter optimization method. Specifically, Keras-Tuner uses a random search to select initial parameter values, then employs Bayesian optimization to evaluate model parameter configurations iteratively. Given the extensive model exploration in Keras-Tuner, nested cross-validation is used to decrease the risk of overfitting. The nested cross-validation scheme partitions the data set into three parts as 80% for training, 10% for validation and 10% for testing. In this way, the first 80% is used to train the model with different parameter configurations, the next 10% is used to validate and compare the performance of these models, and the last 10% is used for final model evaluation.

The hyperparameter optimization is only performed in the TS cross-validation framework to reduce overfitting and computational load, and the optimal hyperparameters obtained are then used in FTS and BFTS. The decision to do so is based on both statistical and practical grounds. From an econometric perspective, the TS training set spans the longest time horizon, thereby providing a richer information set and enabling more efficient and stable hyperparameter selection. In contrast, FTS and BFTS use multiple smaller (and possibly overlapping) training subsets. Running independent hyperparameter searches for each subset would substantially increase the computational cost and complicate the model selection process even further, without obvious gains in terms of reliability. The dataset is organized using a nested cross-validation scheme, ensuring clear separation between the training, validation, and test stages. Within this framework, we consider two different hyperparameter configurations for each dataset: one for the standard LSTM model and one for the Diff-LSTM specification. This enables model-specific tuning while maintaining a consistent evaluation setting. To avoid data leakage, the temporal dependencies are strictly preserved during hyperparameter tuning and model final evaluation. In particular, the temporal ordering of observations is strictly respected, that is, validation folds do not contain any information that is chronologically posterior to the training data used for hyperparameter selection. This ensures that all results reported are based on information sets that would have been available in real-time, thus maintaining the integrity of the forecasting exercise.

In the present study, the LSTM model includes an input layer, two hidden layers, a dropout layer, and two dense layers. Keras-Tuner is used for parameter optimization, which initially requires determining a set of parameter intervals and model attributes. These are as follows:

- To ascertain the optimal activation function for hidden and dense layers, *linear*, *ReLU*, *sigmoid*, and *tanh* activation functions are incorporated into the experimental design.
- To optimize the number of units in hidden and dense layers, a search range of 8 to 512 units, with a step size of 16, is specified for the experimental trials.
- To optimize the dropout rate, a search range of 0.1 to 0.5, with a step size of 0.1, is specified for the experimental trials.
- To determine the optimal learning rate, values of 0.1, 0.01, 0.001, and 0.0001 are provided as a priori information for the experimental trials.
- The experimental trials are conducted utilizing 200 epochs and a batch size of 256.
- The experimental trials are conducted using 500 iterations.

The procedure of finding optimal model parameters with the Keras-Tuner hyperparameter configuration is as follows: First, 500 different model configurations are tested based on pre-defined model attributes and ranges. The best model architecture is chosen according to the lowest RMSE score in the designated validation set. The best model is then trained for 200 epochs and the epoch with the least RMSE is chosen as the optimal epoch. Then, a set of batch sizes (32, 64, 72, 108, 128, 144, and 256) are tested to construct the optimal model architecture. The optimal batch size with the lowest RMSE is chosen. Lastly, to reduce the risk of overfitting, an early stopping callback is also applied.

We chose to use Keras-Tuner with a random search algorithm for hyperparameter selection, rather than trial-and-error or using prior information. The best configuration of standard LSTM and Diff-LSTM models were obtained by choosing different hyperparameters. In other words, each model is used with its best hyperparameters specialized for forecasting the series being analyzed. Although employing identical hyperparameters for

both models would simplify the analysis, it would limit their ability to achieve peak performance individually. Besides, the Diff-LSTM approach uses first-differenced data, changing the structure of the dataset and possibly adding negative values to the series. Consequently, applying the same hyperparameters to both of the original series—which contains no negative values and operates on a larger numerical scale—and the first-order differenced series would result in an unfair comparison. Accordingly, hyperparameters were independently specified for the original series (used in LSTM) and the first-order differenced series (used in Diff-LSTM) within each dataset. The optimized hyperparameters for each model and dataset combinations are presented in Table 3.

An analysis of the hyperparameter selections presented in Table 3 reveals that the optimal hyperparameters for the standard LSTM and Diff-LSTM models differ across almost all series. These variations may be due to characteristics of the data set. The logarithms of closing prices for each index exhibit large, positive values and a distinct trend, whereas the logarithmic differences of the series are trendless and consist of notably smaller, negative values. Furthermore, this disparity in magnitude may affect the computational speed of LSTM units. However, since the primary objective of this study is to compare the peak performance of both models, the underlying causes of these hyperparameter differences were not investigated further.

Table 3 Hyperparameters Selected for LSTM and Diff-LSTM Models According to Datasets

	NYSE Composite		S&*P* 500		NASDAQ Composite	
	LSTM	Diff-LSTM	LSTM	Diff-LSTM	LSTM	Diff-LSTM
Units for Layer 1	136	280	312	472	408	328
Units for Layer 2	280	472	72	488	344	120
Units for Dense 1	16	24	56	16	64	24
Units for Dense 2	1	1	1	1	1	1
Activation Function for Layer 1	*linear*	*sigmoid*	*linear*	*ReLU*	*sigmoid*	*tanh*
Activation Function for Layer 2	*tanh*	*linear*	*tanh*	*tanh*	*linear*	*tanh*
Activation Function for Dense 1	*linear*	*linear*	*linear*	*sigmoid*	*ReLU*	*sigmoid*
Activation Function for Dense 2	*ReLU*	*linear*	*linear*	*tanh*	*linear*	*linear*
Dropout	0.2	0.1	0.2	0.3	0.3	0.1
Learning Rate	0.0001	0.01	0.0001	0.0001	0.0001	0.0001
Epoch	66	9	20	32	39	6
Batch Size	108	108	144	128	144	32
	Russell 2000		Wilshire 5000			
	LSTM	Diff-LSTM	LSTM	Diff-LSTM		
Units for Layer 1	440	472	72	280		
Units for Layer 2	392	456	440	456		
Units for Dense 1	48	40	40	32		
Units for Dense 2	1	1	1	1		
Activation Function for Layer 1	*ReLU*	*tanh*	*ReLU*	*tanh*		
Activation Function for Layer 2	*tanh*	*linear*	*linear*	*tanh*		
Activation Function for Dense 1	*ReLU*	*sigmoid*	*linear*	*linear*		
Activation Function for Dense 2	*linear*	*tanh*	*linear*	*linear*		
Dropout	0.2	0.5	0.1	0.1		
Learning Rate	0.01	0.0001	0.001	0.1		
Epoch	85	171	59	48		
Batch Size	72	256	256	256		

6 Empirical evidence

In the present study, the forecasting performances of the proposed Diff-LSTM and LSTM models are compared using the natural logarithms of the closing values of five major U.S. stock market indices. Model estimates are made based on three cross-validation methods. Model performances are evaluated using multiple metrics: MSE, RMSE, MAE, and MAPE. When making performance comparisons, the LSTM models used in the study are compared with the proposed Diff-LSTM approach. The reason for this is to see how the proposed approach performs compared to the current LSTM methods used in literature. Since Diff-LSTM is based on LSTM forecasting, it is important to compare its performance with LSTM itself. For this purpose, the Diff-LSTM statistics served as a basis, and the percentage change was calculated accordingly[2]. The percentage changes according to all performance criteria are shown in Table 4.

When we compare the performances of the Diff-LSTM and LSTM, Table 4 reveals that Diff-LSTM forecasts perform better across all indices and cross-validations, except for the NYSE Composite index in TS for which LSTM performs only slightly better than Diff-LSTM, by no more than 1.432%. For the remaining indices and cross-validations, on the other hand, Diff-LSTM outperforms LSTM dramatically such that in FTS and BFTS we see that there is a performance increase of almost 100%.

To evaluate whether the LSTM models used in the study perform better or worse than Diff-LSTM, the Diff-LSTM was taken as the benchmark, and the LSTM models were ranked accordingly to provide a vivid visual overview. The results are presented in Table 5 in the order of TS, FTS, and BFTS.

Table 5 shows that Diff-LSTM achieves the best results across all cross-validation settings and for all series, except for the NYSE Composite Index in TS cross-validation. Therefore, it is reasonable to claim that Diff-LSTM demonstrates superior forecasting performance against LSTM across various cross-validation scenarios, supporting the robustness of the proposed method.[3,4,5]

7 Extensions

As an additional exercise, we report the observed and forecast values for the LSTM and Diff-LSTM models under the 10-fold time-series (FTS) cross-validation scheme for all five indices that we employed in Appendix Figs. 3, 4, 5, 6 and 7. In these plots, the blue line denotes the actual index levels, the red line shows the Diff-LSTM forecasts, and the green line is for the forecasts from the standard LSTM model. Visual inspection indicates that

[2] The percentage change is calculated as *[(Criterion(LSTM) - Criterion(Diff-LSTM)) / Criterion(Diff-LSTM)] * 100*. For better understanding, the calculation for MSE is *[(MSE(LSTM) - MSE(Diff-LSTM)) / MSE(Diff-LSTM)] * 100*.

[3] The performance criteria metrics for all models are reported in the Appendix (Table 6). The analyses were repeated using the Day of the Week effect adjusted indices in Table 7. The results do not show any significant difference. The results are provided in the Appendix (Tables 7 and 8).

[4] In addition to LSTM and Diff-LSTM, we also applied two widely used traditional forecasting models, which are the Autoregressive Integrated Moving Average (ARIMA) and Random Walk (RW), as part of our broader research. The corresponding estimates and detailed results are available from the authors upon request. Our main finding is that the proposed Diff-LSTM approach outperforms the traditional LSTM, ARIMA, and RW models across most empirical comparisons in terms of forecasting accuracy and stability. Diff-LSTM consistently ranks among the top performers across multiple indices and cross-validation schemes, underlining its robustness and practical value for financial forecasting.

[5] Even though the optimized Keras-Tuner hyperparameter configurations for LSTM and Diff-LSTM were used to reflect each model's best performance, this also raises the possibility that part of the observed performance gap might be driven by hyperparameter choices rather than by the architecture itself. To address this, we re-estimated all Diff-LSTM specifications using the hyperparameters optimized for the standard LSTM and report the resulting percentage differences. Even under this deliberately conservative design—where Diff-LSTM is "handicapped" by being forced to use hyperparameters tuned for the baseline model—the Diff-LSTM still systematically improves upon the standard LSTM in almost all cases, confirming that its superiority does not hinge on a more favorable hyperparameter search but reflects a genuine gain from the differencing-based specification.

 https://doi.org/10.1007/s00521-026-12369-5

Table 4 Comparison of Diff-LSTM Relative to LSTM as a Percentage

	MSE	RMSE	MAE	MAPE
	Time-series Split (TS)			
NYSE Composite	-1.432%	-0.714%	-1.040%	-1.037%
S&P 500	24.399%	13.051%	21.995%	21.408%
NASDAQ Composite	8.056%	4.112%	7.006%	6.984%
Russell 2000	0.785%	0.393%	0.539%	0.551%
Wilshire 5000	8.803%	4.503%	4.683%	4.628%
	10-Fold Time-series Split (FTS)			
NYSE Composite	99.824%	95.792%	96.318%	86.539%
S&P 500	99.909%	96.902%	97.384%	86.199%
NASDAQ Composite	99.976%	98.397%	98.761%	98.532%
Russell 2000	99.600%	94.003%	94.471%	78.896%
Wilshire 5000	99.649%	94.097%	94.625%	79.947%
	Blocked 5-Fold Time-series Split (BFTS)			
NYSE Composite	98.848%	89.337%	90.619%	70.448%
S&P 500	99.767%	95.279%	95.732%	84.071%
NASDAQ Composite	99.927%	97.439%	97.925%	98.029%
Russell 2000	98.855%	89.478%	90.097%	67.485%
Wilshire 5000	99.565%	93.763%	94.293%	77.159%

Positive values denote superior performance of the Diff-LSTM relative to the LSTM, whereas negative values indicate the opposite

Table 5 Ranking of Models with Respect to Diff-LSTM

	Worse ←←←←←Diff-LSTM→→→→→Better						
	Time-series Split			10-Fold Time-series Split		Blocked 5-Fold Time-series Split	
NYSE Composite							
MSE		**Diff-LSTM**	LSTM	LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**
RMSE		**Diff-LSTM**	LSTM	LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**
MAE		**Diff-LSTM**	LSTM	LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**
MAPE		**Diff-LSTM**	LSTM	LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**
S&P 500							
MSE	LSTM	**Diff-LSTM**		LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**
RMSE	LSTM	**Diff-LSTM**		LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**
MAE	LSTM	**Diff-LSTM**		LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**
MAPE	LSTM	**Diff-LSTM**		LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**
NASDAQ Composite							
MSE	LSTM	**Diff-LSTM**		LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**
RMSE	LSTM	**Diff-LSTM**		LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**
MAE	LSTM	**Diff-LSTM**		LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**
MAPE	LSTM	**Diff-LSTM**		LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**
Russell 2000							
MSE	LSTM	**Diff-LSTM**		LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**
RMSE	LSTM	**Diff-LSTM**		LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**
MAE	LSTM	**Diff-LSTM**		LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**
MAPE	LSTM	**Diff-LSTM**		LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**
Wilshire 5000							
MSE	LSTM	**Diff-LSTM**		LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**
RMSE	LSTM	**Diff-LSTM**		LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**
MAE	LSTM	**Diff-LSTM**		LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**
MAPE	LSTM	**Diff-LSTM**		LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**

Table 6 Performance Criteria Calculated According to the Forecast Results of the Models

	Time-series Split		10-Fold Time-series Split		Blocked 5-Fold Time-series Split	
	LSTM	Diff-LSTM	LSTM	Diff-LSTM	LSTM	Diff-LSTM
NYSE Composite						
MSE	**0.00015450**	0.00015672	0.06554341	**0.00011526**	0.01083081	**0.00012480**
RMSE	**0.01242993**	0.01251864	0.24676581	**0.01038337**	0.10184319	**0.01085974**
MAE	**0.00800606**	0.00808932	0.19710534	**0.00725799**	0.08047717	**0.00754950**
MAPE	**0.00083894**	0.00084764	0.00674295	**0.00090768**	0.00312432	**0.00092331**
S&P 500						
MSE	0.00017042	**0.00012884**	0.11780778	**0.00010713**	0.03802870	**0.00008854**
RMSE	0.01305449	**0.01135070**	0.32545547	**0.01008284**	0.19306496	**0.00911425**
MAE	0.00939703	**0.00733012**	0.26107627	**0.00682914**	0.15396726	**0.00657115**
MAPE	0.00115960	**0.00091135**	0.01102176	**0.00152114**	0.00847158	**0.00134942**
NASDAQ Composite						
MSE	0.00025752	**0.00023678**	0.55129458	**0.00013093**	0.25429844	**0.00018532**
RMSE	0.01604750	**0.01538757**	0.67650029	**0.01084523**	0.49712977	**0.01273207**
MAE	0.01158844	**0.01077654**	0.61584534	**0.00762880**	0.46549936	**0.00965722**
MAPE	0.00123852	**0.00115203**	0.07188658	**0.00105509**	0.06792434	**0.00133851**
Russell 2000						
MSE	0.00021476	**0.00021308**	0.04046854	**0.00016186**	0.02390550	**0.00027361**
RMSE	0.01465474	**0.01459711**	0.19691721	**0.01180845**	0.13747368	**0.01446449**
MAE	0.01157737	**0.01151497**	0.15666340	**0.00866237**	0.11160749	**0.01105224**
MAPE	0.00152562	**0.00151721**	0.00622660	**0.00131406**	0.00520489	**0.00169236**
Wilshire 5000						
MSE	0.00013414	**0.00012233**	0.04171853	**0.00014658**	0.02786690	**0.00012135**
RMSE	0.01158185	**0.01106031**	0.19540417	**0.01153567**	0.16276346	**0.01015179**
MAE	0.00858661	**0.00818450**	0.15684047	**0.00842985**	0.13210280	**0.00753879**
MAPE	0.00080391	**0.00076671**	0.00439565	**0.00088148**	0.00351154	**0.00080206**

The cross-validation results indicated that the values shown in bold yielded the best performance

the blue (actual) and red (Diff-LSTM) lines almost coincide over most of the sample, with only small and short-lived deviations. By contrast, the green LSTM forecasts tend to undershoot the realized paths, especially during sustained upward trends, consistent with the downward bias documented in the main analysis. These graphical results confirm our main empirical findings: Diff-LSTM provides consistently lower MSE, RMSE and MAE in most of the cases and captures better the series dynamics, while the regular LSTM systematically underperforms, especially when the data are highly persistent and trending. In addition to the standard LSTM benchmark, we extend our empirical analysis to include three widely used improved machine learning architectures in financial time series forecasting: GRU (Gated Recurrent Unit), ATT-LSTM (Attention-based Long Short-Term Memory), and CNN-LSTM (Convolutional Neural Network - Long Short-Term Memory) within the TS as an additional exercise [27, 35, 46]. The comparison of the forecasting performance of Diff-LSTM against these three additional alternative models for TS is provided in Appendix Table 9. Across all indices and performance metrics, Diff-LSTM generally outperforms these alternative models, with the only partial exception being the NYSE Composite index, where performance differences remain very small.

We evaluate our baseline using four popular performance metrics (MSE, RMSE, MAE and MAPE) but these measures, although informative, are not formal statistical tests of superior predictive performance. In contrast to many studies that rely on only one or two criteria, our use of four complementary metrics already provides a more comprehensive assessment of forecasting performance; nevertheless, we additionally implement the Diebold–Mariano (DM) test to evaluate whether the observed differences in accuracy are statistically significant between LSTM and Diff-LSTM. The DM statistics are reported in Appendix Table 10, which is the analog of Table 4, and they consistently support our main conclusion that Diff-LSTM outperforms the LSTM models in a statistically meaningful manner, thereby confirming the robustness of our findings. The only statistically insignificant

Table 7 Performance Criteria Calculated According to the Forecast Results of the Models with Day of the Week Effect-Free Series

	Time-series Split		10-Fold Time-series Split		Blocked 5-Fold Time-series Split	
	LSTM	Diff-LSTM	LSTM	Diff-LSTM	LSTM	Diff-LSTM
NYSE Composite						
MSE	0.00026429	**0.00015111**	0.97136090	**0.00009504**	0.22161405	**0.00012250**
RMSE	0.01625712	**0.01229260**	0.81894963	**0.00933697**	0.39035041	**0.01075687**
MAE	0.01291334	**0.00794174**	0.78768454	**0.00650026**	0.38274342	**0.00742058**
MAPE	0.00134904	**0.00083164**	0.08864684	**0.00080934**	0.04603380	**0.00090375**
S&P 500						
MSE	0.00015812	**0.00012925**	0.11353899	**0.00010878**	0.03638326	**0.00008736**
RMSE	0.01257475	**0.01136873**	0.32308083	**0.01016925**	0.18856014	**0.00906947**
MAE	0.00902874	**0.00739069**	0.25909709	**0.00695902**	0.14973449	**0.00657421**
MAPE	0.00112178	**0.00091719**	0.01005991	**0.00153241**	0.00573005	**0.00134484**
NASDAQ Composite						
MSE	0.00024856	**0.00024523**	1.55532495	**0.00017141**	0.16680666	**0.00019715**
RMSE	0.01576566	**0.01565988**	1.09078534	**0.01227090**	0.39074956	**0.01307518**
MAE	0.01106049	**0.01098903**	1.03712524	**0.00863057**	0.34669979	**0.00989773**
MAPE	0.00118112	**0.00117350**	0.13169710	**0.00118725**	0.04834046	**0.00137318**
Russell 2000						
MSE	0.00022790	**0.00020645**	0.04112413	**0.00015813**	0.03116164	**0.00027410**
RMSE	0.01509629	**0.01436853**	0.19870408	**0.01148750**	0.16234928	**0.01427261**
MAE	0.01193869	**0.01132321**	0.15808307	**0.00835629**	0.13790522	**0.01094353**
MAPE	0.00157161	**0.00149105**	0.00444851	**0.00126056**	0.00956906	**0.00167118**
Wilshire 5000						
MSE	0.00013868	**0.00012408**	0.03783518	**0.00013199**	0.02502106	**0.00013355**
RMSE	0.01177623	**0.01113905**	0.18671866	**0.01086925**	0.15474131	**0.01062628**
MAE	0.00882464	**0.00825064**	0.14967849	**0.00761721**	0.12510975	**0.00789511**
MAPE	0.00082570	**0.00077283**	0.00289853	**0.00079546**	0.00228459	**0.00083734**

The cross-validation results indicated that the values shown in bold yielded the best performance

outcome arises for the Russell 2000 series under TS cross-validation, for which Table 4 shows a percentage performance difference of less than 1% between the two models. This finding implies that, in this particular case, Diff-LSTM performs on par with the standard LSTM rather than strictly dominating it. Overall, the evidence in Table 10 supports the superior predictive ability of Diff-LSTM across indices and cross-validation schemes.

As for the optimal hyperparameter configuration of the standard LSTM and Diff-LSTM, the use of distinct hyperparameters for each model was justified earlier to avoid limiting the models' forecast ability and achieving peak performance. However, to demonstrate the superiority of the Diff-LSTM, we compared the performance of the standard LSTM and the Diff-LSTM using identical hyperparameters (those of the standard LSTM). This way, the effect of different hyperparameter configurations on each model's forecast performance can be observed and, to some extent, explained. The performance comparisons are reported in Appendix Table 11. The Diff-LSTM still demonstrates superior forecasting performance despite being disadvantaged by using hyperparameters chosen specifically for the standard LSTM. Furthermore, the improvements in the MSE and RMSE criteria observed for the NYSE Composite data in Table 11 highlight the risk of overfitting, which may result from using hyperparameters optimized for the standard LSTM rather than the Diff-LSTM model. Thus, this exercise revealed that the Diff-LSTM still performs better than the standard LSTM even when employed with non-optimal hyperparameters.

Lastly, assessing the performance of our specification under extreme market conditions is an important issue to address. In our empirical design, the FTS and BFTS schemes inherently evaluate the models on multiple non-overlapping test windows, which include several high-volatility episodes and market crashes in U.S. equity

Table 8 Model Ranking of Results According to Diff-LSTM with Day of the Week Effect-Free Series

	Worse ←←←←←Diff-LSTM→→→→→Better					
	Time-series Split		10-Fold Time-series Split		Blocked 5-Fold Time-series Split	
NYSE Composite						
MSE	LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**
RMSE	LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**
MAE	LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**
MAPE	LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**
S&P 500						
MSE	LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**
RMSE	LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**
MAE	LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**
MAPE	LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**
NASDAQ Composite						
MSE	LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**
RMSE	LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**
MAE	LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**
MAPE	LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**
Russell 2000						
MSE	LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**
RMSE	LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**
MAE	LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**
MAPE	LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**
Wilshire 5000						
MSE	LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**
RMSE	LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**
MAE	LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**
MAPE	LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**	LSTM	**Diff-LSTM**

The cross-validation results indicated that the values shown in bold yielded the best performance

indices over 1928–2024. The above-reported out-of-sample metrics and Diebold–Mariano statistics already reflect performance across both tranquil and turbulent regimes. However, to address the issue more explicitly, we replicated the TS-based analysis by restricting the test period to the COVID-19 crash window, with the training sample ending on 19 February 2020, when the S&P 500 peaked just before losing roughly one-third of its value over the following weeks. The results for this subsample remain consistent with our main findings and confirm that Diff-LSTM continues to outperform the standard LSTM during this extremely high volatility episode.

Taken together, these extension analyses show that the superior performance of the Diff-LSTM is not an artifact of any particular metric, cross-validation design, benchmark architecture, or sample period. Diff-LSTM dominates the standard LSTM visually (through prediction–vs–actual plots), quantitatively (through multi-metric error comparisons and Diebold–Mariano statistics), and structurally (when confronted with alternative GRU, ATT-LSTM, and CNN-LSTM benchmarks), and it continues to do so even during the COVID-19 crash window, where volatility and trend reversals are most pronounced. The joint evidence suggests that the advantages of the differencing and reconstructing series are economically meaningful, statistically significant and robust to challenging market conditions, confirming Diff-LSTM as a reliable and practically relevant improvement over classical LSTM-based forecasting approaches.

8 Conclusion

This study introduces the Diff-LSTM approach to enhance the predictive power of Long Short-Term Memory (LSTM) networks by addressing LSTM activation functions' inherent insensitivity to time series data with large values. The modified approach is to model the log-differences of a series and use the differences together with

 https://doi.org/10.1007/s00521-026-12369-5



Fig. 3 Observed and Forecasts Values for the NASDAQ Composite Index: A Comparison of LSTM and Diff-LSTM Models (for FTS)

lagged observed values to forecast the levels of the original series. To prove the effectiveness of this approach, we conduct a comprehensive comparative analysis to compare its performance with a standard LSTM model. This incorporated three distinct cross-validation schemes (TS, FTS, and BFTS), five major US financial indices (NASDAQ Composite, NYSE Composite, Russell 2000, S&P 500 and Wilshire 5000), and four separate performance criteria (MAE, MAPE, MSE and RMSE), culminating in a total of 60 distinct comparisons to demonstrate the improvements in the forecast performance of the Diff-LSTM method thoroughly.

The comparison analyses give two important results. First, the Diff-LSTM outperformed the standard LSTM in 93% of the comparisons. Second, the proposed method is stable across various cross-validation schemes, whereas the baseline LSTM model is not. Namely, the standard LSTM performance drops significantly when switching

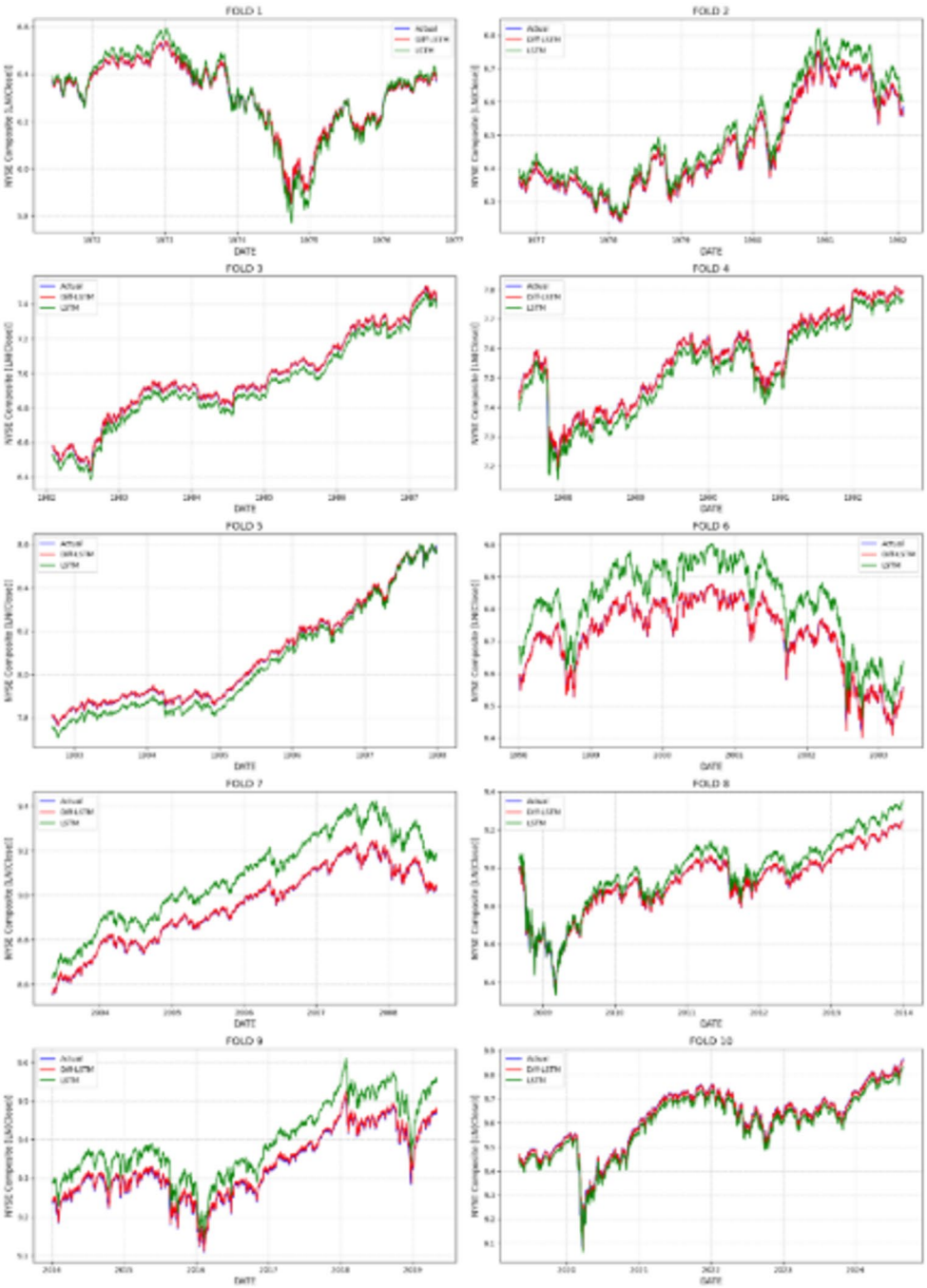

Fig. 4 Observed and Forecasts Values for the NYSE Composite Index: A Comparison of LSTM and Diff-LSTM Models (for FTS)

from the traditional TS cross-validation to other cross-validations with fragmented datasets (FTS and BFTS), while the proposed Diff-LSTM is not affected by this degradation and shows even some improvement in some cases.

Diff-LSTM is designed as a relatively lightweight extension of the standard LSTM: it preserves the same basic architecture and training pipeline, and its additional operations (differencing and related transformations) are implemented at the pre-processing and input-representation level rather than through a substantial increase in network depth or parameter count. Therefore, training and inference times are of the same order of magnitude as an ordinary LSTM, making the method practically feasible for real-time or high-frequency financial applications. Regarding interpretability, the underlying intuition of Diff-LSTM is explicitly econometric since the model

 https://doi.org/10.1007/s00521-026-12369-5

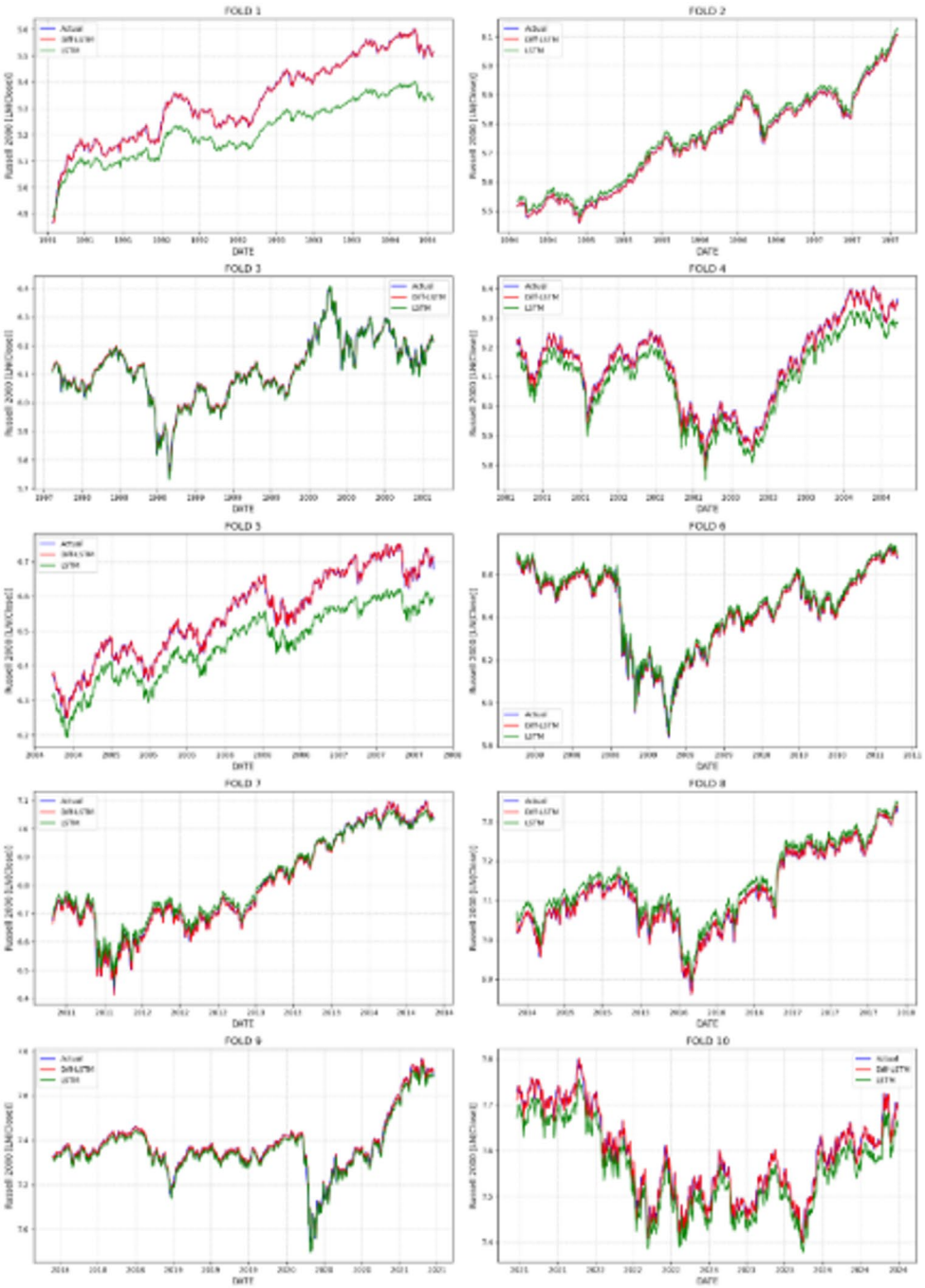

Fig. 5 Observed and Forecasts Values for the Russell 2000 Index: A Comparison of LSTM and Diff-LSTM Models (for FTS)

operates on differenced series, i.e., it focuses on changes rather than levels, consistent with the standard approach in time-series econometrics and facilitating the connection between the learned dynamics and return behavior, volatility and shock propagation. This structure provides an economically sensible interpretation of the model's outputs, without the opacity usually associated with more complex deep architectures, which typically use multiple attention layers or heavily engineered feature extractors.

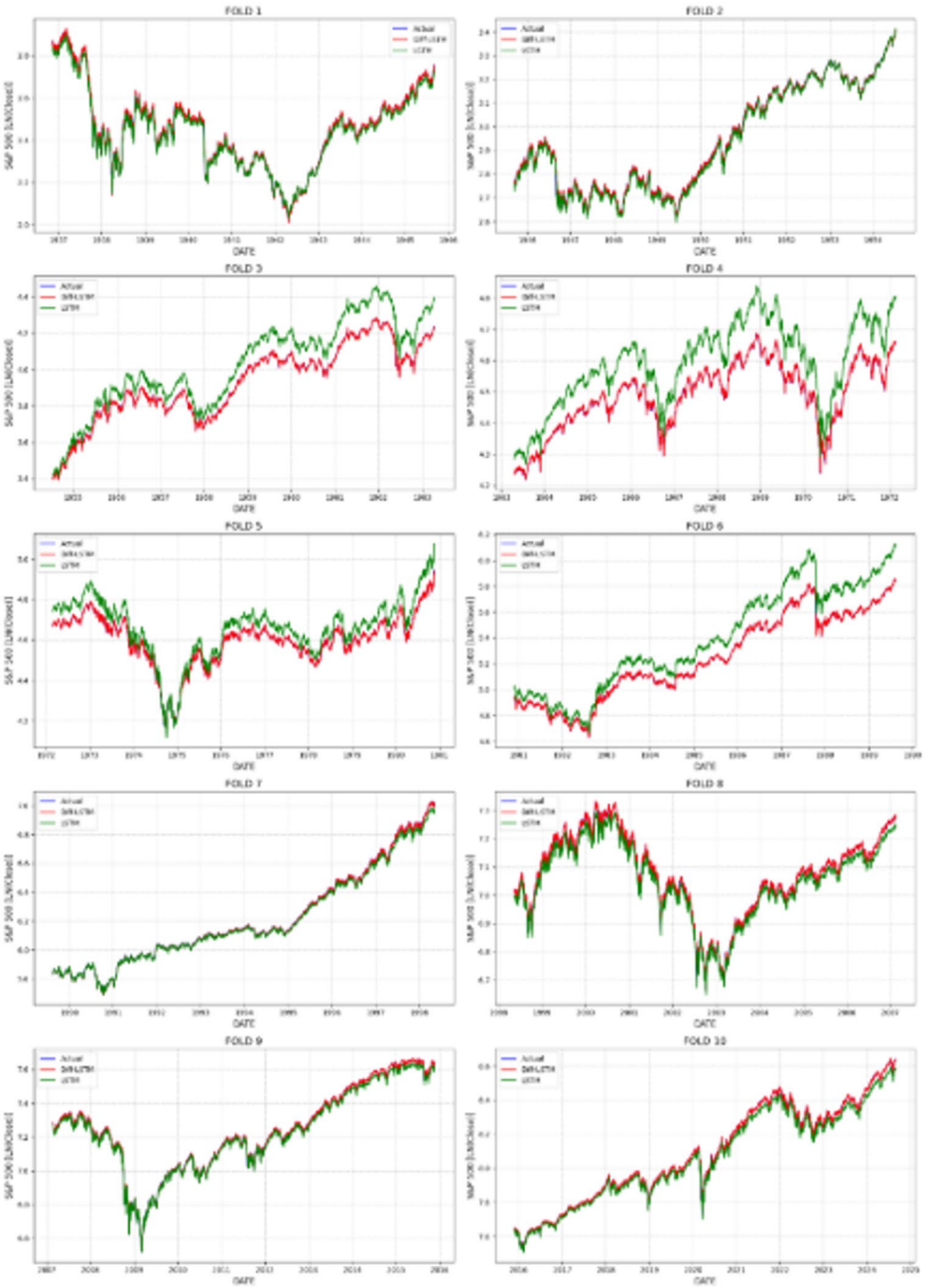

Fig. 6 Observed and Forecasts Values for the S&P 500 Index: A Comparison of LSTM and Diff-LSTM Models (for FTS)

 https://doi.org/10.1007/s00521-026-12369-5

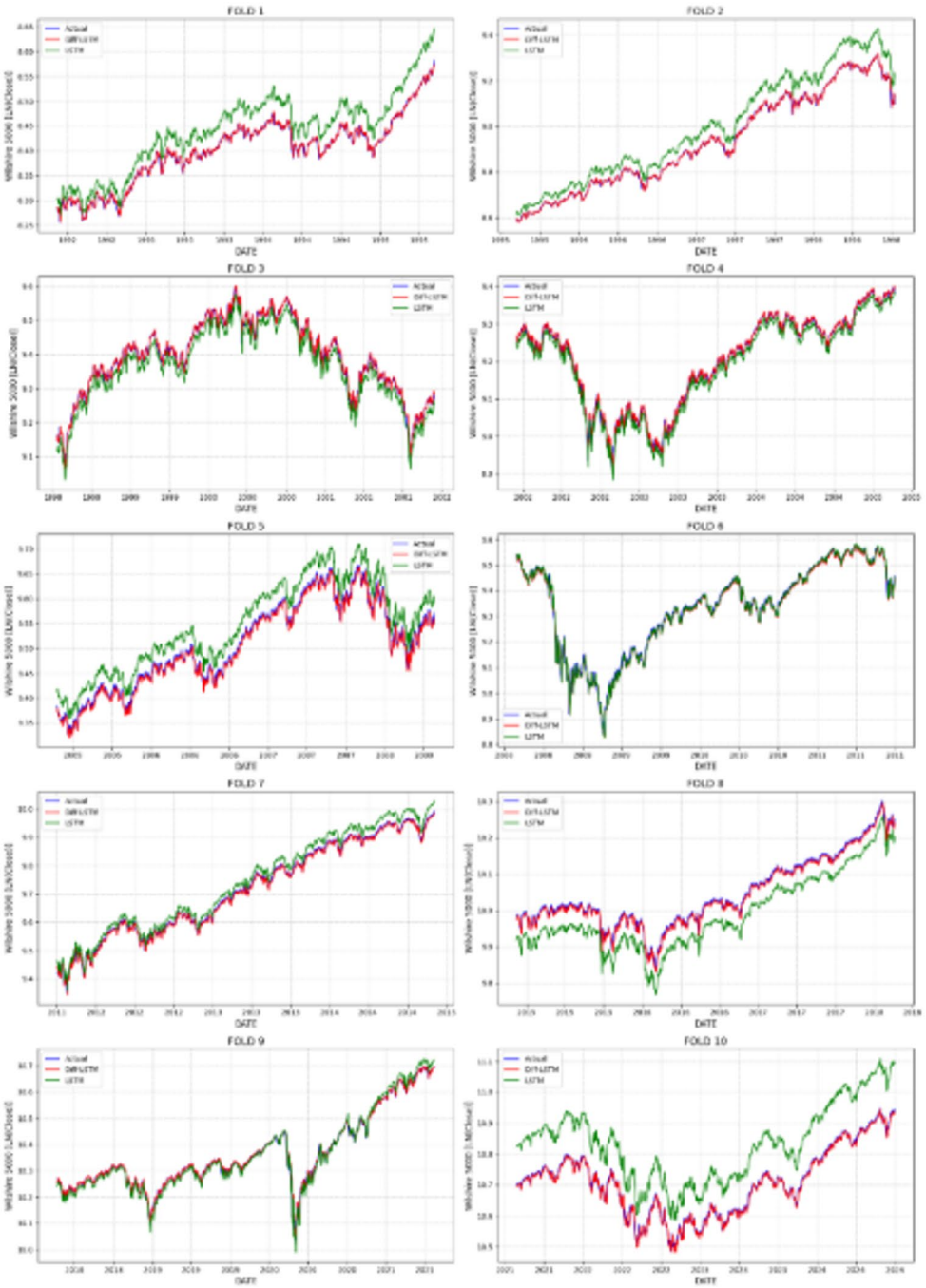

Fig. 7 Observed and Forecasts Values for the Wilshire 5000 Index: A Comparison of LSTM and Diff-LSTM Models (for FTS)

Table 9 A Comparison of Forecasting Performance: Diff-LSTM Against 4 Alternative Models (for TS)

NYSE Composite	LSTM	Diff-LSTM	GRU	ATT-LSTM	CNN-LSTM
MSE	0.00015450	0.00015319*	0.00015411	0.00015450	0.00024942
RMSE	0.01242993	0.01237696*	0.01241419	0.01242994	0.01579315
MAE	0.00800606	0.00800985	0.00799856*	0.00800606	0.01169905
MAPE	0.00083894	0.00083929	0.00083815*	0.00083894	0.00122064
S&P 500					
MSE	0.00017042	0.00012882*	0.00202196	0.00017042	0.00178621
RMSE	0.01305449	0.01134977*	0.04496623	0.01305450	0.04226355
MAE	0.00939703	0.00735328*	0.03977752	0.00939704	0.03564397
MAPE	0.00115960	0.00091421*	0.00486342	0.00115960	0.00439097
NASDAQ Composite					
MSE	0.00025752	0.00024577*	0.00024683	0.00025399	0.01277079
RMSE	0.01604750	0.01567704*	0.01571075	0.01593706	0.11300790
MAE	0.01158844	0.01097705*	0.01101037	0.01141534	0.10884616
MAPE	0.00123852	0.00117347*	0.00117694	0.00122013	0.01151519
Russell 2000					
MSE	0.00021476	0.00021039*	0.00023406	0.00021298	0.00022580
RMSE	0.01465474	0.01450477*	0.01529888	0.01459375	0.01502670
MAE	0.01157737	0.01144155*	0.01205611	0.01153103	0.01195984
MAPE	0.00152562	0.00150756*	0.00159003	0.00151939	0.00157429
Wilshire 5000					
MSE	0.00013414	0.00012279*	0.00012657	0.00013414	0.00012532
RMSE	0.01158185	0.01108105*	0.01125020	0.01158180	0.01119464
MAE	0.00858661	0.00821459*	0.00836471	0.00858658	0.00830566
MAPE	0.00080391	0.00076946*	0.00078338	0.00080391	0.00077788

An asterisk (*) denotes the best performance for each evaluation metric

Table 10 Diebold-Mariano Test Results

	NYSE Composite	S&*P* 500	NASDAQ Composite	Russell 2000	Wilshire 5000
	Time-series Split (TS)				
DM-Statistic	3.083***	-14.005***	-6.856***	-0.299	-4.387***
	10-Fold Time-series Split (FTS)				
DM-Statistic (1. Fold)	-24.206***	-24.845***	-27.518***	-41.648***	-46.925***
DM-Statistic (2. Fold)	-39.456***	-26.191***	-89.552***	-40.509***	-39.418***
DM-Statistic (3. Fold)	-97.011***	-67.299***	-235.661***	-14.973***	-27.511***
DM-Statistic (4. Fold)	-51.382***	-107.178***	-178.111***	-42.747***	-16.719***
DM-Statistic (5. Fold)	-54.558***	-66.869***	-29.183***	-55.156***	-50.263***
DM-Statistic (6. Fold)	-103.345***	-57.749***	-110.283***	-15.258***	5.877***
DM-Statistic (7. Fold)	-83.570***	-23.677***	-586.990***	-12.672***	-30.242***
DM-Statistic (8. Fold)	-44.633***	-47.205***	-928.975***	-28.143***	-82.096***
DM-Statistic (9. Fold)	-84.472***	-29.933***	-1391.488***	-16.146***	-9.675***
DM-Statistic (10. Fold)	-41.321***	-41.100***	-1287.310***	-32.118***	-10.522***
	Blocked 5-Fold Time-series Split (BFTS)				
DM-Statistic (1. Fold)	-16.783***	-28.240***	-47.062***	-112.560***	-32.429***
DM-Statistic (2. Fold)	-25.553***	-59.310***	-41.133***	-29.124***	-32.088***
DM-Statistic (3. Fold)	-6.825***	-44.968***	-26.345***	-2.010**	-10.866***
DM-Statistic (4. Fold)	-59.374***	-79.441***	-53.616***	-8.821***	-10.206***
DM-Statistic (5. Fold)	-21.179***	-161.584***	-57.779***	-17.354***	-6.697***

***, **, * symbols indicate that the test statistics exceeded the critical value at the 1%, 5%, and 10% significance levels, respectively. Here, the null hypothesis (H_0) argues that the two models have equal predictive power, i.e., there is no difference between their respective error series. Conversely, the alternative hypothesis (H_1) suggests that a difference exists, and one model is superior to the other

Table 11 Comparison of Diff-LSTM Relative to LSTM as a Percentage (with LSTM's Hyperparameters)

	MSE	RMSE	MAE	MAPE
	Time-series Split (TS)			
NYSE Composite	0.851%	0.426%	-0.047%	-0.042%
S&P 500	24.412%	13.059%	21.749%	21.161%
NASDAQ Composite	4.564%	2.308%	5.276%	5.253%
Russell 2000	2.036%	1.023%	1.173%	1.184%
Wilshire 5000	8.461%	4.324%	4.333%	4.285%
	10-Fold Time-series Split (FTS)			
NYSE Composite	99.831%	95.892%	96.443%	86.978%
S&P 500	99.907%	96.870%	97.327%	85.974%
NASDAQ Composite	99.969%	98.193%	98.607%	98.348%
Russell 2000	99.515%	93.297%	93.812%	76.171%
Wilshire 5000	99.691%	94.485%	95.219%	82.152%
	Blocked 5-Fold Time-series Split (BFTS)			
NYSE Composite	98.878%	89.471%	90.811%	71.045%
S&P 500	99.768%	95.270%	95.740%	84.060%
NASDAQ Composite	99.922%	97.364%	97.872%	97.968%
Russell 2000	98.853%	89.342%	89.939%	66.670%
Wilshire 5000	99.527%	93.509%	94.085%	76.411%

9 Appendix

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s00521-026-12369-5.

Funding Open access funding provided by Bilkent University. No funding was received.

Declarations

Competing interest The authors declare that they have no competing interests.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Abu-Mostafa YS, Atiya AF (1996) Introduction to financial forecasting. Appl Intell 6:205–213. https://doi.org/10.1007/BF00126626
2. Aguilera PA, Fernández A, Reche F, Rumí R (2010) Hybrid Bayesian network classifiers: Application to species distribution models. Environ Model Softw 25(12):1630–1639
3. Andrews DW (1993) Exactly median-unbiased estimation of first order autoregressive/unit root models. Econometrica: J Econometric Soc 61:139–165
4. Armstrong JS (2001) Principles of forecasting: A handbook for researchers and practitioners. Springer, pp 443–472
5. Armstrong JS, Collopy F (1992) Error measures for generalizing about forecasting methods: Empirical comparisons. Int J Forecast 8(1):69–80
6. Bartlett MS (1946) On the theoretical specification and sampling properties of autocorrelated time-series. Supplement J Royal Stat Soc 8(1):27–41

7. Booms TL, Huettmann F, Schempf PF (2010) Gyrfalcon nest distribution in Alaska based on a predictive GIS model. Polar Biol 33:347–358
8. Bordes A, Glorot X, Weston J, Bengio Y (2012) Joint learning of words and meaning representations for open-text semantic parsing. In Artificial Intelligence and Statistics (pp. 127–135)
9. Breiman L, Spector P (1992) Sub model selection and evaluation in regression: The X-random case. Int Stat Rev / Revue Int de Statistique 60(3):291–319
10. Chatfield C (1988) Apples, oranges and mean square error. Int J Forecast 4(4):515–518
11. Dahl G, Ranzato MA, Mohamed AR, Hinton GE (2010) Phone recognition with the mean-covariance restricted Boltzmann machine. Adv Neural Inf Process Syst, 23
12. Deboeck GJ (1994) Trading on The Edge: Neural, Genetic, and Fuzzy Systems for Chaotic Financial Markets, vol 39. Wiley
13. Dickey DA (1976) Estimation and Hypothesis Testing in Nonstationary Time Series. Ph.D. thesis, Iowa State University
14. Dickey DA, Fuller WA (1981) Likelihood ratio statistics for autoregressive time series with a unit root. Econometrica 49:1057–1072
15. El-Amir H, Hamdy M (2019) Deep Learning Pipeline: Building a Deep Learning Model with TensorFlow. Apress: Berkeley, CA, USA
16. Fang W, Chen Y, Xue Q (2021) Survey on research of RNN-based spatio-temporal sequence prediction algorithms. J Big Data 3(3):97
17. Fildes R (1993) The evaluation of extrapolative forecasting methods: International Journal of Forecasting, 8 (1), 81–98 (June 1992). Long Range Planning, 26(1), 151
18. Fischer T, Krauss C (2018) Deep learning with long short-term memory networks for financial market predictions. Eur J Oper Res 270(2):654–669
19. Fuller WA (2009) Introduction to Statistical Time Series. Wiley
20. Gülmez B (2023) Stock price prediction with optimized deep LSTM network with artificial rabbits optimization algorithm. Expert Syst Appl 227:120346
21. Hafshejani MS, Mansouri N (2025) Enhancing stock market prediction with LSTM: A review of recent developments and comparative analysis. Arch Comput Methods Eng. https://doi.org/10.1007/s11831-025-10370-0
22. Han Z, Zhu X, Su Z (2024) Forecasting maritime and financial market trends: Leveraging CNN-LSTM models for sustainable shipping and China's financial market integration. Sustainability 16(22):9853
23. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780
24. Huck N (2009) Pairs selection and outranking: An application to the S&P 500 index. Eur J Oper Res 196(2):819–825
25. Hurwicz L (1950) Generalization of the concept of identification. Stat Inference Dynamic Economic Models 10:245–257
26. Jothi LS (2025) Time series analysis using LSTM networks and its application to financial forecasting. In: Derbali AMS (ed) Artificial Intelligence for Financial Risk Management and Analysis. IGI Global Scientific Publishing, pp 19–58
27. Ju J, Liu FA (2021) Multivariate time series data prediction based on ATT-LSTM network. Appl Sci 11(20):9373
28. Kendall MG (1954) Note on bias in the estimation of autocorrelation. Biometrika 41(3–4):403–404
29. Khang PQ, Hernes M, Kuziak K, Rot A, Gryncewicz W (2020) Liquidity prediction on Vietnamese stock market using deep learning. Procedia Comput Sci 176:2050–2058
30. Kim B-J, Nam I-W (2025) A review of hybrid LSTM models in smart cities. Processes 13(7):2298. https://doi.org/10.3390/pr13072298
31. Kohzadi N, Boyd MS, Kermanshahi B, Kaastra I (1996) A comparison of artificial neural network and time series models for forecasting commodity prices. Neurocomputing 10(2):169–181
32. Krizhevsky A, Sutskever I, Hinton GE (2017) ImageNet classification with deep convolutional neural networks. Commun ACM 60(6):84–90
33. Le Breton A, Pham DT (1989) On the bias of the least squares estimator for the first order autoregressive process. Ann Inst Stat Math 41:555–563
34. Li Q, Kamaruddin N, Yuhaniz SS, Al-Jaifi HAA (2024) Forecasting stock prices changes using long-short term memory neural network with symbolic genetic programming. Sci Rep 14(1):422
35. Lu W, Li J, Li Y, Sun A, Wang J (2020) A CNN-LSTM-based model to forecast stock prices. Complexity 2020(1):6622927
36. Makridakis S, Andersen A, Carbone R, Fildes R, Hibon M, Lewandowski R, Winkler R (1982) The accuracy of extrapolation (time series) methods: Results of a forecasting competition. J Forecast 1(2):111–153
37. Manowska A, Rybak A, Dylong A, Pielot J (2021) Forecasting of natural gas consumption in Poland based on ARIMA-LSTM hybrid model. Energies 14(24):8597
38. Marriott FHC, Pope JA (1954) Bias in the estimation of autocorrelations. Biometrika 41(3/4):390–402
39. Mateńczuk K, Kozina A, Markowska A, Czerniachowska K, Kaczmarczyk K, Golec P, Dyvak M (2021) Financial time series forecasting: Comparison of traditional and spiking neural networks. Procedia Comput Sci 192:5023–5029
40. Mendes JFB (2019) Forecasting Bitcoin Prices: ARIMA vs LSTM [Master's thesis]. ISCTE-Instituto Universitario de Lisboa

41. Olah C (2015) August 27 Understanding LSTM networks. colah's blog. https://colah.github.io/posts/2015-08-Understanding-LSTMs/
42. Orcutt GH, Irwin JO (1948) A study of the autoregressive nature of the time series used for Tinbergen's model of the economic system of the United States, 1919–1932. J Royal Stat Soc Ser B (Methodological) 10(1):1–53
43. Orcutt GH, Winokur HS Jr (1969) First order autoregression: Inference, estimation, and prediction. Econometrica 37:1–14
44. Pantula SG, Fuller WA (1983) Mean estimation bias in least squares estimation of autoregressive processes. J Econ 27(1):99–121
45. Patterson J, Gibson A (2017) Deep learning: A Practitioner's Approach. O'Reilly Media
46. Pirani M, Thakkar P, Jivrani P, Bohara MH, Garg D (2022) A comparative analysis of ARIMA, GRU, LSTM and BiLSTM on financial time series forecasting. In 2022 IEEE International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE) (pp. 1–6). IEEE
47. Roondiwala M, Patel H, Varma S (2017) Predicting stock prices using LSTM. Int J Sci Res (IJSR) 6(4):1754–1756
48. Rumelhart DE, Hinton GE, Williams RJ (1986) Learning representations by back-propagating errors. Nature 323(6088):533–536
49. Sastry ASR (1951) Bias in estimation of serial correlation coefficients. Sankhyā: Indian J Stat 11:281–296
50. Schmidhuber J (1993) Habilitation thesis: System modeling and optimization. Credit assignment across the equivalent of 1200 layers in an RNN
51. Shaman P, Stine RA (1988) The bias of autoregressive coefficient estimators. J Am Stat Assoc 83(403):842–848
52. Sherstinsky A (2020) Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. Physica D 404:132306
53. Siami-Namini S, Tavakoli N, Namin AS (2018) A comparison of ARIMA and LSTM in forecasting time series. In 17th IEEE International Conference on Machine Learning and Applications (ICMLA). IEEE. https://doi.org/10.1109/ICMLA.2018.00227
54. Sims CA, Uhlig H (1991) Understanding unit rooters: A helicopter tour. Econometrica 59:1591–1599
55. Wang J, Hong S, Dong Y, Li Z, Hu J (2024) Predicting stock market trends using LSTM networks: Overcoming RNN limitations for improved financial forecasting. J Comput Sci Softw Appl 4(3):1–7
56. Wang SW, Huang CY (2026) A hybrid SVR-based framework for cryptocurrency price forecasting and strategy backtesting. Appl Artif Intell 40(1). https://doi.org/10.1080/08839514.2026.2612793
57. Wang Y, Zhang H, Huang B, Lin Z, Pang C (2025) LSTM stock prediction model based on blockchain. High-Confidence Comput 5:100316
58. White JS (1961) Asymptotic expansions for the mean and variance of the serial correlation coefficient. Biometrika 48(1–2):85–94
59. Xiong R, Nichols EP, Shen Y (2015) Deep learning stock volatility with Google domestic trends. arXiv preprint arXiv:1512.04916
60. Yamamoto T, Kunitomo N (1984) Asymptotic bias of the least squares estimator for multivariate autoregressive models. Ann Inst Stat Math 36(3):419–430
61. Yang S, Yu X, Zhou Y (2020) LSTM and GRU neural network performance comparison study: Taking Yelp review dataset as an example. In 2020 International Workshop on Electronic Communication and Artificial Intelligence (IWECAI) (pp. 98–101)
62. You Y (2024) Forecasting stock price: A deep learning approach with LSTM and hyperparameter optimization. Highlights Sci Eng Technol 85:328–338
63. Yu Y, Si X, Hu C, Zhang J (2019) A review of recurrent neural networks: LSTM cells and network architectures. Neural Comput 31(7):1235–1270. https://doi.org/10.1162/neco_a_01199
64. Zhang Q, Yang LT, Chen Z, Li P (2018) A survey on deep learning for big data. Inform Fusion 42:146–157
65. Zhao Y, Hasan YA (2013) Machine learning algorithms for predicting roadside fine particulate matter concentration level in Hong Kong Central. Comput Ecol Softw 3(3):61

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Authors and Affiliations

Levent Gul[1] · Ebru Yuksel Haliloglu[2] · Hande Sevgi[3] · Nukhet Dogan[4] · M. Hakan Berument[5]

✉ M. Hakan Berument
berument@bilkent.edu.tr

Levent Gul
levent.gul@ogr.hbv.edu.tr; leventgul@kku.edu.tr

Ebru Yuksel Haliloglu
ebruyuksel@gazi.edu.tr

Hande Sevgi
hande.sevgi@ufuk.edu.tr

Nukhet Dogan
nukhet.dogan@hbv.edu.tr

[1] Department of Econometrics, Kirikkale University, Kirikkale 71450, Turkey

[2] Department of Industrial Engineering, Gazi University, Ankara 06560, Turkey

[3] Department of International Trade and Finance, Ufuk University, Ankara 06805, Turkey

[4] Department of Econometrics, Ankara Haci Bayram Veli University, Ankara 06500, Turkey

[5] Department of Economics, Bilkent University, Ankara 06800, Turkey