Hongjoong KIM, PhD
hongjoong@korea.ac.kr
Korea University, South Korea

Kyoung-Sook MOON, PhD (corresponding author)
ksmoon@gachon.ac.kr
Gachon University, South Korea

Optimal Data Construction in Supervised Machine Learning for Financial Prediction

Abstract. *Despite the widespread adoption of machine learning and deep learning in financial market forecasting, achieving satisfactory results remains a persistent challenge due to the inherent complexities, nonlinearity, and uncertainties in financial data. This study addresses this challenge by introducing two innovative methods in supervised machine learning for feature and label construction. Features are derived from the graphical representation of price data to capture inherent patterns, and the target variable definition is grounded in data momentum, enabling predictions beyond the confines of the training data. These methodological advancements not only enhance regression accuracy, but also expedite model training by facilitating predictions of unobserved values during training. Empirical analysis, employing various financial market datasets and neural network models, demonstrates a substantial improvement in prediction accuracy and efficiency.*

Keywords: *prediction of financial market, deep learning, smoothing of financial time series, data construction.*

JEL Classification: C15, C45, C53, C63.

1. Introduction

Accurately predicting financial market dynamics remains a formidable challenge due to the inherent uncertainties, nonstationarity, and nonlinearity of financial data, compounded by substantial market noise. Despite these challenges, machine learning and deep learning techniques have demonstrated promise in financial data prediction, encompassing methods such as linear regression, decision trees, support vector machines, ensemble methods like random forest and gradient boosting, as well as two-stage methods involving various combinations (Atsalakis and Valavanis, 2009; Sezer et al., 2020; Moon and Kim, 2019, 2023). Notably, the long short-term memory (LSTM) method has gained wide acceptance for predicting financial time-series data (Fischer and Krauss, 2018; Chen and Ge, 2019; Nabipour et al., 2020; Su et al., 2021; Jung and Choi, 2021).

Despite the effectiveness of these approaches, the presence of noise in financial data remains a limiting factor, diminishing the precision and efficacy of prediction methods (Hassani et al., 2010). Traditionally, to address this challenge, widely adopted smoothing methods such as the simple moving average (SMA) or exponentially weighted moving average (EWMA) are employed to mitigate noise in

DOI: 10.24818/18423264/58.1.24.01

financial data (Ellis and Parbery, 2005; Babu and Reddy, 2014). Although these moving average techniques are straightforward and commonly used, a noteworthy issue arises from the time lag associated with the reflection of historical data (Raudys et al., 2013). To overcome this limitation, alternative denoising methods have been proposed. Notable among them are the wavelet transform (Sun and Meinl, 2012; He and He, 2019; Wu et al., 2021), and the Fourier transform (Song et al., 2021), which aim to enhance the robustness of financial data by reducing noise and preserving essential information for improved predictive accuracy.

We present an innovative data representation methodology that aims at eliminating noise and improving the efficacy of supervised learning algorithms when applied to financial data. This introduced approach comprises two distinct components: the first method focuses on noise reduction in features, while the second method is dedicated to constructing a target variable. Our methodology demonstrates superior predictive outcomes, as elucidated in Chapter 4, showcasing the effectiveness of these techniques in refining the precision and reliability of financial data analysis through supervised learning algorithms.

To address noise

in financial data, we propose a graph-based smoothing method known for its intuitiveness, simplicity, and speed. Graphing stock prices can reveal severe vibrations, and our smoothing method effectively suppresses these vibrations when applied with a thick pen, as illustrated. Additionally, we advocate for using the momentum of values as the target variable for swift and accurate regression. Learning momentum, as opposed to the value itself, enables predictions of rising or falling markets not evident in the training data.

Combining these two methods with recurrent neural network (RNN) or long short-term memory (LSTM) models facilitates rapid and accurate predictions of future values for noisy financial variables, ultimately contributing to increased profitability in portfolio management. The experimental validation in Section 4 confirms the improved accuracy and efficiency compared to alternative feature construction and targeting methods.

The subsequent sections of this paper are structured as follows. Section 2 introduces a novel algorithm that incorporates graph-based smoothing and defines the momentum-based difference as a target variable. In Section 3, we detail the dataset and the deep learning models employed. Section 4 offers a comprehensive summary of empirical results obtained through the application of the proposed algorithm to diverse financial datasets. Concluding the paper, Section 5 provides insights and conclusions drawn from the study's findings.

2. Methods

2.1 Graph-based feature construction

We first explain the transformation from an oscillatory time-series data to a graph-based smooth data in a few steps. Let $S_i \coloneqq S(t_i)$ be the actual value of the time series at time $t_i, i = 1,2,...$, and let $X = \{S_1, S_2, ...\}$ be the whole set of S_i's. Here, the value S_i represents the price of an asset, such as a stock, an index, or an exchange rate, in the financial market. As shown in Figure 1, we construct subsets of the values starting from S_i to the past n values, $\boldsymbol{S_i} = [S_{i-n+1}, \dots, S_{i-1}, S_i]$, $i = 1,2, \dots$.

The objective of this study is to predict value $\widehat{S_i} := S_{i+p}$ after p units in time from t_i by learning these subsets. In ordinary supervised machine learning methods, $\boldsymbol{S_i}$ and $\widehat{S_i}$ are referred to as *feature* and *target*, respectively, and dataset $\{(\boldsymbol{S_i}, \widehat{S_i})\}$ is used for learning. However, such construction of the dataset may not be appropriate in finance because of the complexity, nonlinearity, and uncertainty of the financial data. Thus, in this study, we propose new methods to construct the feature and the target so that supervised learning becomes accurate.

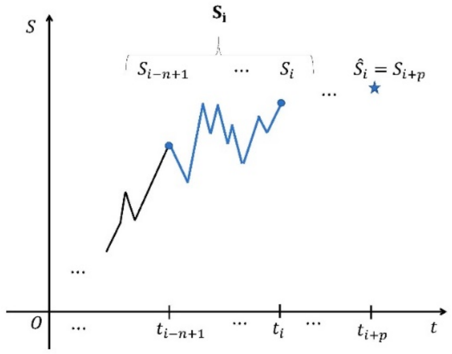


Figure 1. Feature and target of the financial data
Source: Illustration by authors.

Given the history values, $\boldsymbol{S_i}$, at $t = t_i$, $\boldsymbol{S_i}$ can be provided to a network for training by using it as it is without any smoothing. This is called *Raw* data in this study. Because financial assets with various market uncertainties are volatile, the raw values of features $\boldsymbol{S_1}, \boldsymbol{S_2}, \dots$ in many real-world applications are oscillatory and various averaging techniques, such as the SMA or EWMA, are applied to derive smooth features. The SMA with a window length of w is defined as

$$S_i^{SMA} = \frac{1}{\omega}\left(S_{i-\omega+1} + S_{i-\omega+2} + \cdots + S_i\right) \tag{1}$$

and the EWMA with parameter α is defined as

$$S_i^{EWMA} = \frac{S_i + (1-\alpha)S_{i-1} + (1-\alpha)^2 S_{i-2} + \cdots + (1-\alpha)^i S_0}{1 + (1-\alpha) + (1-\alpha)^2 + \cdots + (1-\alpha)^i} \tag{2}$$

Thus, the SMA redefines $\boldsymbol{S_i}$ with

$$\boldsymbol{S_i} = \{S_{i-n+1}^{SMA}, \cdots, S_{i-1}^{SMA}, S_i^{SMA}\},$$

And the EWMA redefines $\boldsymbol{S_i}$ with

$$\boldsymbol{S_i} = \{S_{i-n+1}^{EWMA}, \cdots, S_{i-1}^{EWMA}, S_i^{EWMA}\}.$$

In this study, $\omega = 5$ for the SMA and $\alpha = 0.2421$ for the EWMA are used.

As an alternative method to effectively reduce oscillations, we propose a graph-based smoothing method for each of $\boldsymbol{S_i}$'s as follows. The first step is to draw the graph of each $\boldsymbol{S_i}$, as depicted in Figure 2a. Next, $2k + 1$ vertical grid lines are introduced at $t = t_{i,j}$:

$$t_{i,j} = t_{i-n+1} + j\,\Delta t, j = 0,1,2,\dots,2k$$

with $\Delta t = (t_i - t_{i-n+1})/2k$, and $2k + 1$ horizontal grid lines are introduced at $y = y_{i,j}$:

$$y_{i,j} = \bar{\boldsymbol{S}_i} + j\Delta y, j = -k, -k+1, \dots, k$$

as displayed in Figure 2b. Here, $\bar{\boldsymbol{S}_i}$ is the empirical mean of $\boldsymbol{S_i}$:

$$\bar{\boldsymbol{S}_i} = \frac{1}{n}\sum_{j=0}^{n-1} S_{i-j} \tag{3}$$

and

$$\Delta y = \frac{2z_\beta \sigma/\sqrt{n}}{2k}$$

where σ approximates the *population* standard deviation of universal set X and z_β is the value for the $100(1 - \beta)\%$ confidence level; for instance, for *95%* confidence level, $z_\beta = 1.96$. Those $2k + 1$ horizontal and vertical lines partition area

$$[t_{i-n+1}, t_i] \times \left[\bar{\boldsymbol{S}_i} - z_\beta\frac{\sigma}{\sqrt{n}}, \bar{\boldsymbol{S}_i} + z_\beta\frac{\sigma}{\sqrt{n}}\right]$$

of the graph into $(2k) \times (2k)$ uniform cells. Here, $k = 10$ is used. In the third step, the region, R_j , $j = 1, 2, \dots, 2k$, is identified such that the graph of $\boldsymbol{S_i}$ passes through (dark gray in Figure 2c). For example, in Figure 2c, R_1 consists of 3 cells, R_2 consists of 4 cells, etc. Then, for each $j = 1, 2, \dots, 2k$, let $s_{i,j}$ be the center of the selected cells for $[t_{i,j-1}, t_{i,j}]$:

$$s_{i,j} = \frac{1}{2}\left(\min_{k\in R_j} y_{i,k} + \max_{k\in R_j} y_{i,k}\right) \tag{4}$$

The red dots in Figure 2d show these centers, $\{s_{i,1}, s_{i,2}, \dots, s_{i,2k}\}$, which define the graph-based smooth data for $\boldsymbol{S_i}$.

This new graph-based smoothing can be applied to express financial data with severe oscillations, as in Figure 2a, as smooth data, as in Figure 2d, while retaining the original fluctuation pattern. This method is intuitive, simple, and fast. In addition, the number of the values in the feature at t_i is reduced from $|\boldsymbol{S_i}| = n$ to $2k$. Therefore, the computation cost for model training is reduced and the speed of machine learning is increased.

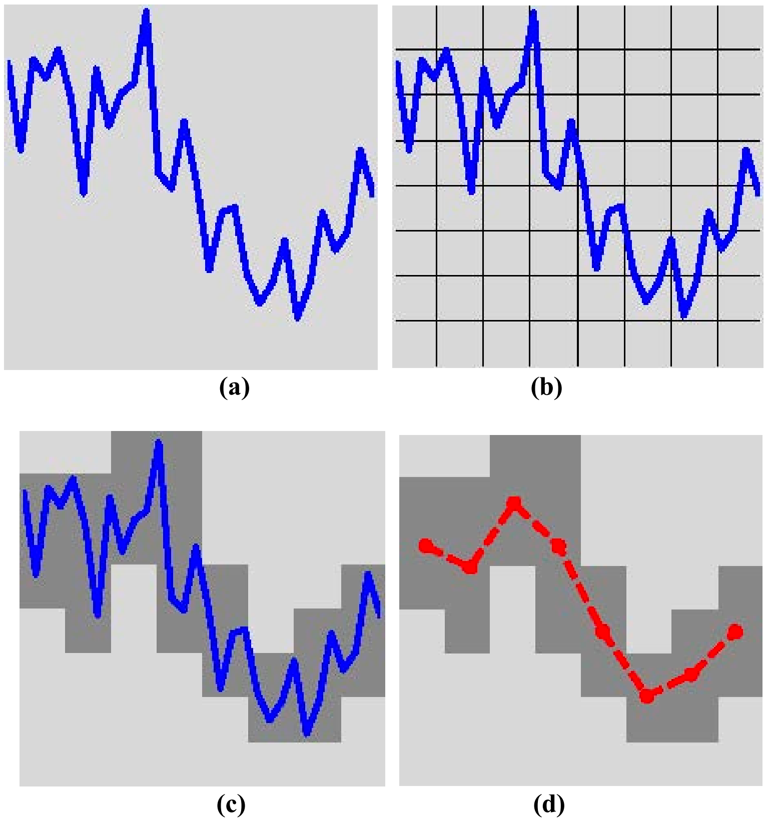


Figure 2. Graph-based transformation from an oscillatory time series to smooth data: (a) graph of an oscillatory raw time series, (b) graph with grids, (c) graph and the cells through which the graph passes, (d) graph-based smooth data approximating the raw data
Source: Illustration by authors.

Table 1 summarises the values used as the feature depending on the smoothing scheme.

Table 1. Values used as the feature depending on the smoothing scheme

Smoothing scheme	Values used as the feature
Raw	$\{S_{i-n+1}, \dots, S_{i-1}, S_i\}$
SMA	$\{S_{i-n+1}^{SMA}, \cdots, S_{i-1}^{SMA}, S_i^{SMA}\}$ from (1)
EWMA	$\{S_{i-n+1}^{EWMA}, \cdots, S_{i-1}^{EWMA}, S_i^{EWMA}\}$ from (2)
Graph-based	$\{s_{i,1}, s_{i,2}, \dots, s_{i,2k}\}$ from (4)

Source: Values used by authors.

2.2 Momentum-based target construction

When the feature construction procedure is completed, an appropriate target variable is introduced. For the target variable, the actual value in p days, $\widehat{S_i} = S_{i+p}$, is used as the target at $t = t_i$ in many studies if the value in p days is predicted.

As indicated by extensive empirical experiments, the prediction values of the test data are closely related with the *range* of the target variable values of the training data and the predicted values may be inaccurate if the values of the target variable of the test data are beyond the range from the training data. Thus, the usage of $\widehat{S_i} = S_{i+p}$ as the target may not be appropriate unless the range from the training data includes the range from the test data.

Let us define δS_i by the following equation:

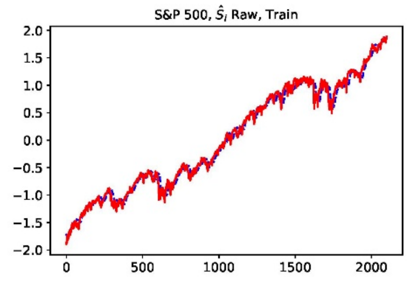
$$\delta S_i \equiv S_{i+p} - \boldsymbol{S}_i^{avg} \tag{5}$$

where $\boldsymbol{S}_i^{avg}$ is the arithmetic average of the values in $\boldsymbol{S_i}$, as indicated in Table 1. In finance, momentum is the tendency for rising asset prices to rise further and falling prices to keep falling, and thus difference $S_{i+p} - \boldsymbol{S}_i^{avg}$ can be considered as a numerical metric for estimating the momentum. δS_i represents the amount of deviation between the average value of the feature and the value to be predicted. We propose the use of δS_i as a new target variable alternative to $\widehat{s_i}$. Because of the subtraction of $\boldsymbol{S}_i^{avg}$ in (5), the ranges from training, validation, and test data are centralised and δS_i is expected to improve the prediction accuracy and be a better target variable.

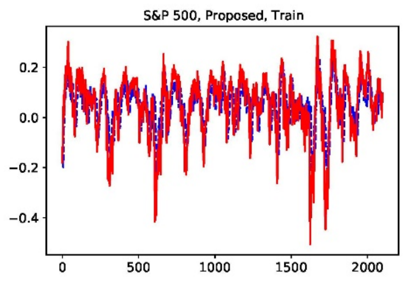
We compare the prediction results of S&P 500 from January 1, 2009 to October 29, 2021, using LSTM. The description of the empirical tests is explained in detail in Section 3 and Section 4. Figure 3a shows the actual target values, $\widehat{S_i}$ (red), and their predictions, $\widetilde{S_i}$ (blue), for the training data when the *raw* values of the Close prices of S&P 500 are used as the feature to predict the future value in 10 days. The range of the actual target value, $\widehat{S_i}$, during the training period is $[-1.5, 2.0]$, and the prediction, $\widetilde{S_i}$, is close to $\widehat{S_i}$.

Figure 3c shows $\widehat{S_i}$ and $\widetilde{S_i}$ values for the validation data. The range of the values of $\widehat{S_i}$ in this case is $[1.5, 4.0]$, which is slightly deviated from the range of the training data, and the corresponding prediction, $\widetilde{S_i}$, exhibits some errors. The predictions corresponding to $\widehat{S_i}$ outside $[-1.5, 2.0]$ of the training data deviate considerably from the actual values. When the test data are considered in Figure 3e, the range of $\widehat{S_i}$ values is $[1, 6]$ and the predictions corresponding to $\widehat{S_i}$outside $[-1.5, 2.0]$ deviate substantially from the actual values. As the range of the test data deviates considerably from that of the training data, the errors become worse.

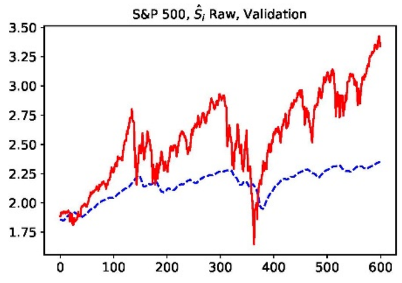
Figure 3b, Figure 3d, and Figure 3f present the results when δS_i in (5) is used as a target variable. Because $\boldsymbol{S}_i^{avg}$is subtracted when the target variable is defined in (5), the ranges of $\widetilde{S_i}$ from the validation in Figure 3d and the test data in Figure 3f are mostly included in the range from the training data and the prediction accuracy is improved, that is, the predictions for the validation and test data are close to the actual values. This new target variable, δS_i, in (5) is effective particularly when the range of the predicted variable from the training data differs from the range from the test data, and the effects of $\delta S_i$are numerically validated in a more comprehensive manner in Section 4.



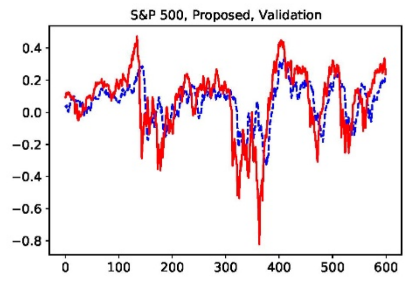
(a)



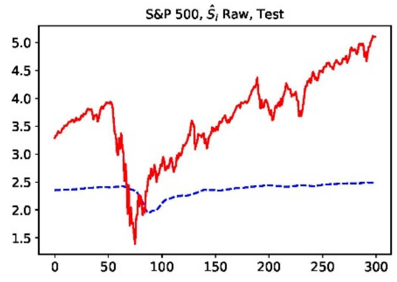
(b)



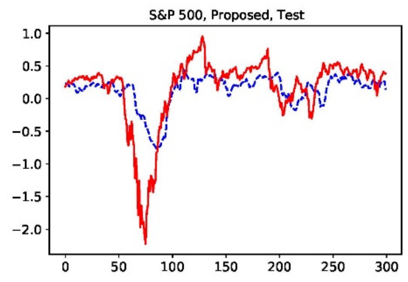
(c)



(d)



(e)



(f)

Figure 3. Comparison of target variables $\widehat{S_i}$ ((a), (c), and (e)) and δS_i ((b), (d), and (f)) using the S&P 500 values from January 1, 2009 to October 29, 2021: (a) and (b) training data, (c) and (d) validation data, and (e) and (f) test data
Source: Calculation made by authors.

Table 2 summarises the target variable considered in this study.

Table 2. Target variables used in this study

Target variable	Definition
$\widehat{S}_i$	S_{i+p}
δS_i	$S_{i+p} - \boldsymbol{S}_i^{avg}$

Source: Notations used by authors.

The proposed algorithm and its effectiveness can be summarised as follows.

- We propose two methods for machine learning: one for the construction of the feature and the other for the construction of a target variable. We combine these two methods with neural network algorithms.
- The feature construction algorithm smoothens oscillatory data and reduces the amount of data in the feature, so that the learning speed of the proposed algorithm increases. The new definition of the target variable improves the accuracy of the prediction.
- The proposed methods can be easily combined with various machine learning or deep learning methods.
- Regression results can be applied to the valuation and applications of financial variables, such as portfolio management or pricing of financial derivatives.

3. Data and learning models

3.1 Dataset

The data are constructed with four datasets from the global market, namely, S&P 500, Dow Jones, Gold, and Russel 2000, and two datasets from the Korean market, namely, KOSDAQ and USD/KRW currency. Each dataset contains opening, high, low, and closing prices and the total traded quantity, and the daily closing prices from January 1, 2009 to October 29, 2021, are used. Table 3 shows the statistics of the datasets.

Table 3. Statistics of four datasets from the global market (S&P 500, Dow Jones, Gold, and Russel 2000) and two datasets from the Korean market (KOSDAQ and USD/KRW currency)

	S&P500	Dow Jones	GOLD	Russel 2000	KOSDAQ	USD/KRW
Count	3230	3230	3203	3230	3163	3230
Mean	2124.85	18687.01	1380.07	1187.99	632.36	1137.65
Std	883.26	7041.83	253.71	443.91	149.99	66.83
Min	676.53	6547.05	806.70	343.26	339.76	999.83
25%	1345.07	12720.73	1215.05	813.04	514.71	1098.79
50%	2041.95	17515.58	1308.20	1164.27	607.37	1128.54
75%	2729.69	24753.82	1586.40	1503.22	700.48	1167.90
Max	4596.42	35756.88	2051.50	2360.17	1060.00	1571.40

Source: Calculation made by authors.

Among the data, 2100 values from each set are used for training, 600 values for validation, and 300 values for the test. Because the datasets considered exhibit distinct price ranges and deep learning is typically efficient when the data are in a common range, the following standardisation procedure of the data is performed:

$$S_i^{modified} = \frac{S_i - \mu_{tr}}{\sigma_{tr}} \tag{6}$$

where μ_{tr} and σ_{tr} are the average and standard deviation, respectively, of S_i's in the *training* data. This standardised data, $S_i^{modified}$, is used instead of S_i and denoted as S_i for notational simplicity. $\boldsymbol{S_i}$ is then constructed with a window size of 40.

3.2 Deep learning models

The RNN is an artificial neural network in which hidden nodes are connected by directed edges to form a directed cycle, as depicted in Figure 4a. The RNN is a model suitable for processing data that appears sequentially, such as voice and text. However, when the distance between the relevant information and the point where the information is used is large, the RNN gradually decreases the gradient during backpropagation, which considerably reduces the learning ability. To overcome this problem, LSTM is designed, as shown in Figure 4b. In this structure, the cell-state is added to the hidden state of the RNN (Géron, 2019; Sherstinsky, 2020).

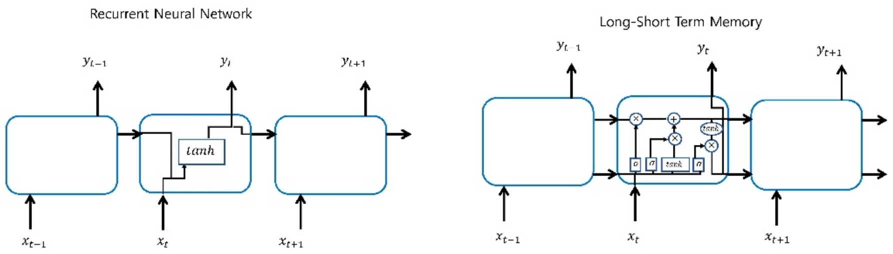


Figure 4. Comparison of the stricture of (a) RNN and (b) LSTM
Source: Illustration by authors.

Two types of neural networks are constructed for training. One network consists of two layers of a fully connected RNN, each of which is followed by dropout. A densely connected neural network layer exists at the end. The dimensionality of the output space for the RNN is 20, and the fraction of the input units to drop for dropout is 0.2. The other network consists of two layers of LSTM instead of RNN layers, and the dimensionality of the output space for LSTM is set to 20 as well. The RNN and LSTM are implemented with Keras in Tensorflow 2.4 on Intel i7-11700 3.60 GHz with NVIDIA GeForce RTX 3090. The RNN and LSTM networks are trained for 60 epochs (see more details in Goodfellow (2016) and Géron, (2019)). Algorithm 1 summarises the prediction algorithm of the proposed method.

4. Empirical Tests

We analyse the effects of various techniques for the feature construction and define the target variable. We compare each method in terms of two metrics. In the first evaluation, the accuracy of the prediction is measured, and the mean squared error (MSE) and mean absolute error (MAE) are used to evaluate prediction accuracy. The MSE measures the average of squares of errors between the actual value and the predicted value:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\left(\widehat{S_i} - \widetilde{S_i}\right)^2 \qquad (7)$$

and the MAE measures the average of the absolute errors between the actual value and the predicted value as follows:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}\left|\widehat{S_i} - \widetilde{S_i}\right| \qquad (8)$$

where $\widehat{S_i} = S_{i+p}$ is the actual value to be predicted at time $t = t_i$ and $\widetilde{S_i}$ is the prediction at $t = t_i$. N is the number of predictions. In the second evaluation, the model training speed is measured by checking the CPU time. The CPU time is measured in seconds. Here, $p = 10$ is used in this study. That is, the value on the $(t + 10)^{th}$day is predicted at day t. Table 4 summarises the parameters used for the deep learning in this study.

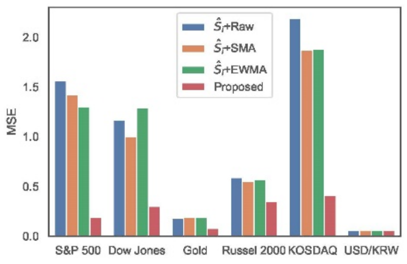
Table 4. Parameters for deep learning

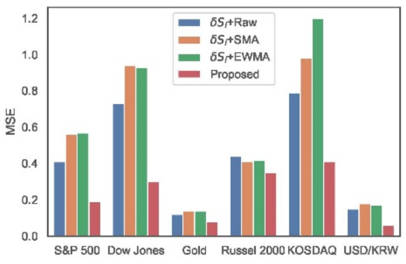
the number of train data	2100
the number of validation data	600
the number of test data	300
the window size of Si	40
the units in time (p) to be predicted	10
the dimensionality of the output space for RNN and LSTM	20
the fraction of the input units to drop	0.2
epochs for RNN and LSTM	60

Source: Parameters used by authors.

We first evaluate the prediction accuracy. The proposed method is compared with six combinations in terms of constructing feature $\boldsymbol{S_i}$ (Raw, SMA or EWMA) and defining the target variable ($\widehat{S_i}$ or δS_i).

Figure 5 shows the MSE from the RNN. *Raw* implies that the raw data is used without any smoothing technique. SMA and EWMA present the results from the SMA and the EWMA, respectively. *Proposed* shows the results from the graph-based feature with δS_i as the target variable. The MSE value from the proposed graph-based feature and momentum-based target is compared with the MSE values from the raw-, SMA-, and EWMA-based features with $\widehat{S_i}$ as a target variable (Figure 5a) and with those from the raw-, SMA-, and EWMA-based features with δS_i as a target variable (Figure 5b). The proposed method results in the smallest error regardless of the dataset.





(a) (b)

Figure 5. MSE from the RNN (a) $\widehat{S_\iota}$ -based prediction vs proposed method and (b) δS_i -based prediction vs. proposed method

Source: Calculation made by authors.

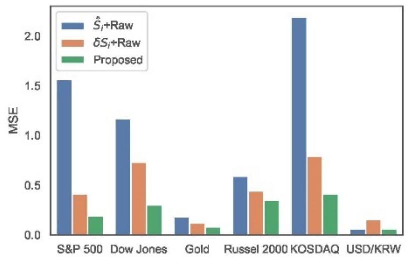


Figure 6. MSEs of raw data in $\widehat{S_\iota}$ -based prediction vs. δS_i -based prediction vs. the proposed method from the RNN

Source: Calculation made by authors.

Figure 6 presents a comparison of the result of the proposed method with those of raw-based features. The MSE of the proposed method is smaller than those of $\widehat{S_\iota}$ -Raw or δS_i-Raw case in all datasets. Similar patterns are observed in SMA-based and EWMA-based features as well, and the plots are omitted.

Table 5 presents the prediction accuracy measured in terms of the MSE (7) for all situations and datasets. When S&P 500 dataset is trained over the RNN network, the errors are between 1.30 and 1.56 if $\widehat{S_\iota}$ is used as a target variable and between 0.41 and 0.57 if δS_i is used as a target variable. The application of δS_i reduces prediction error, and similar results are observed in the other datasets. SMA and EWMA are ineffective except for the marginal gain in S&P 500 and KOSDAQ over the RNN. When the target variable, δS_i , is combined with the graph-based smoothing from the proposed method, considerable improvement is observed and the error of S&P 500 decreases to 0.19, which is only 15% compared with $\widehat{S_\iota}$ -based estimation and 46% compared with δS_i-based estimation. In case of Dow Jones, the MSE from $\widehat{S_\iota}$ over the RNN is between 1.00 and 1.29, which decreases to between 0.73 and 0.94 with δS_i and decreases further to 0.30 with the proposed method. Similar results are observed in Russel 2000 and KOSDAQ.

In case of Gold, the proposed method results in an MSE of only 0.08, which is smaller than the errors from $\widehat{S_i}$ or δS_i , but the errors from both $\widehat{S_i}$ and δS_i are sufficiently small as well. This phenomenon results from the ranges of the target variables. The range of $\widehat{S_i}$ of the test data is mostly included in the range of the training data in case of Gold or USD/KRW datasets; this leads to excellent accuracy even with $\widehat{S_i}$ and δS_i. However, the error from the proposed method is still smaller than either of them.

The results from the LSTM network are similar to those from the RNN. The prediction error from δS_i is smaller than that from $\widehat{S_i}$, and the error from the proposed method is substantially smaller than those from $\widehat{S_i}$ or δS_i.

Table 5. Prediction MSE from (top) the RNN and (bottom) the LSTM network

RNN(MSE)		S&P500	Dow Jones	Gold	Russel 2000	KOSDAQ	USD/KRW
$\widehat{S_i}$	Raw	1.56	1.17	0.18	0.59	2.19	0.06
	SMA	1.42	1.00	0.19	0.55	1.87	0.06
	EWMA	1.30	1.29	0.19	0.57	1.88	0.06
δS_i	Raw	0.41	0.73	0.12	0.44	0.79	0.15
	SMA	0.56	0.94	0.14	0.41	0.98	0.18
	EWMA	0.57	0.93	0.14	0.42	1.20	0.17
Proposed		**0.19**	**0.30**	**0.08**	**0.35**	**0.41**	**0.06**

LSTM(MSE)		**S&P500**	**Dow Jones**	**Gold**	**Russel 2000**	**KOSDAQ**	**USD/KRW**
$\widehat{S_i}$	Raw	2.18	2.19	0.39	0.68	1.48	0.09
	SMA	2.26	2.29	0.41	0.72	1.37	0.10
	EWMA	2.17	2.32	0.43	0.66	1.41	0.10
δS_i	Raw	0.30	4.30	0.21	1.55	0.96	0.14
	SMA	0.30	4.24	0.27	1.36	1.63	0.13
	EWMA	0.29	4.30	0.23	1.42	1.57	0.13
Proposed		**0.16**	**0.28**	**0.11**	**0.30**	**0.39**	**0.06**

Source: Calculation made by authors.

Table 6 presents the MAE from the RNN and the LSTM networks. Similarly to the MSE, the proposed method with the graph- and momentum-based data construction improves the prediction accuracy in both the RNN and the LSTM networks.

Table 6. Prediction MAE from (top) the RNN and (bottom) the LSTM network

RNN(MAE)		**S&P500**	**Dow Jones**	**Gold**	**Russel 2000**	**KOSDAQ**	**USD/KRW**
$\widehat{S_i}$	Raw	1.15	0.99	0.37	0.57	1.26	0.17
	SMA	1.10	0.91	0.38	0.56	1.17	0.19
	EWMA	1.04	1.04	0.39	0.56	1.17	0.18
δS_i	Raw	0.44	0.64	0.27	0.40	0.72	0.33
	SMA	0.56	0.76	0.29	0.40	0.80	0.37

RNN(MAE)		**S&P500**	**Dow Jones**	**Gold**	**Russel 2000**	**KOSDAQ**	**USD/KRW**
	EWMA	0.57	0.75	0.28	0.41	0.89	0.36
Proposed		**0.29**	**0.32**	**0.22**	**0.40**	**0.52**	**0.18**

LSTM(MAE)		**S&P500**	**Dow Jones**	**Gold**	**Russel 2000**	**KOSDAQ**	**USD/KRW**
$\widehat{S_i}$	Raw	1.35	1.37	0.53	0.60	1.02	0.23
	SMA	1.38	1.41	0.54	0.62	1.01	0.25
	EWMA	1.35	1.41	0.54	0.60	1.03	0.24
δS_i	Raw	0.30	1.90	0.38	0.94	0.80	0.31
	SMA	0.31	1.91	0.43	0.87	1.10	0.29
	EWMA	0.32	1.92	0.40	0.89	1.08	0.28
Proposed		**0.27**	**0.31**	**0.25**	**0.37**	**0.50**	**0.19**

Source: Calculation made by authors.

Table 7. Computation time (in seconds) from (top) the RNN and (bottom) the LSTM network

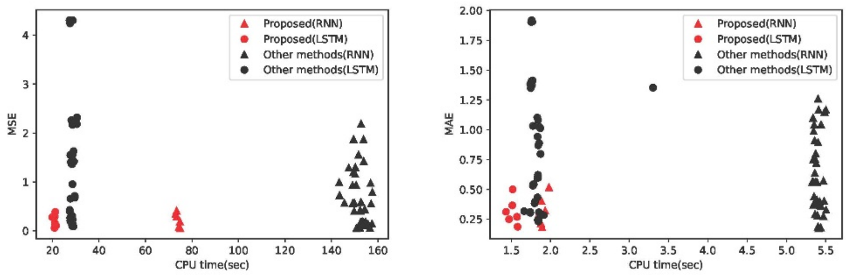
RNN (second)		**S&P500**	**Dow Jones**	**Gold**	**Russel 2000**	**KOSDAQ**	**USD/KRW**
$\widehat{S_i}$	Raw	151.61	150.17	152.85	152.73	152.73	150.73
	SMA	153.90	143.34	154.63	157.07	153.78	152.24
	EWMA	150.29	147.47	153.21	149.86	149.53	156.57
δS_i	Raw	152.56	143.48	154.15	154.34	157.58	157.44
	SMA	149.18	149.06	152.94	19.77	156.82	153.78
	EWMA	145.58	150.60	155.77	152.32	149.41	152.44
Proposed		**74.84**	**73.45**	**74.33**	**72.93**	**73.43**	**74.96**

LSTM (second)		**S&P500**	**Dow Jones**	**Gold**	**Russel 2000**	**KOSDAQ**	**USD/KRW**
$\widehat{S_i}$	Raw	30.52	28.55	27.66	29.54	28.49	29.03
	SMA	28.12	30.37	27.40	29.60	28.14	28.76
	EWMA	28.59	30.56	27.43	27.75	27.92	28.55
δS_i	Raw	28.53	28.76	27.63	27.64	28.61	28.79
	SMA	28.49	27.59	27.62	28.44	29.06	28.32
	EWMA	28.49	27.31	28.74	29.26	28.80	28.71
Proposed		**20.97**	**19.92**	**21.58**	**20.98**	**21.22**	**20.91**

Source: Calculation made by authors.

Table 7 presents the corresponding CPU time (in seconds) for the computation. The proposed method requires only 75 s with the RNN, while the other methods require approximately 150 s. When machine learning is employed to train the LSTM network, only 20 s are required with the proposed method and approximately 30 s with the other methods. Because the proposed method reduces the size of the feature through the graph-based construction, the computational cost is reduced to approximately a half for the RNN and two thirds for the LSTM network. As indicated by the accuracies in Table 5 and Table 6 and the computational costs in Table 7, LSTM requires less computational time for similar accuracies. Thus, learning over the LSTM network is more efficient than learning over the RNN.

Figure 7 presents a comparison of the computation cost versus the prediction accuracy when the RNN and the LSTM network are used for machine learning. The prediction accuracy is measured in terms of the MSE in Figure 7a and the MAE in Figure 7b. Even when the difference in the network structure is considered, Figure 7 reveals that the LSTM network is more effective when considering the prediction accuracy and computational cost. Because the computation for the RNN is expensive and the difference in the prediction accuracy is not large enough, training over the LSTM network is preferred. These experimental results show that the proposed graph- and momentum-based data construction method improves the prediction accuracy and computational efficiency.



(a) (b)

Figure 7. Computational cost vs. the prediction error when RNN and LSTM network are used. (a) CPU time vs. MSE (b) CPU time vs. MAE
Source: Calculation made by authors.

5. Conclusions

In conclusion, we presented two effective algorithms: a graph-based feature construction method tailored for handling noisy financial data and a novel momentum-based technique to define a target variable, enhancing the prediction accuracy of machine learning methods. The observed improvements in both accuracy and efficiency were substantiated through applications in asset price prediction, employing diverse datasets and neural networks. In particular, the versatility of the proposed method extends its applicability to various financial domains beyond asset price prediction. This includes areas such as derivative pricing, bitcoin trading, and robo-advisors, all of which represent pivotal topics of ongoing research in the financial realm. This research contributes significant insights and practical applications, paving the way for further advancements in the utilisation of these algorithms across diverse financial contexts.

Acknowledgements: *This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2021R1F1A1054766) and the Korea government (MSIT) (NRF-2018R1D1A1B07050046).*

References

[1] Atsalakis, G.S., Valavanis, K.P. (2009), *Surveying stock market forecasting techniques - part II: Soft computing methods. Expert Systems with Applications*, 36, 5932-5941.

[2] Babu, C.N., Reddy, B.E. (2014), *A moving-average filter based hybrid ARIMA–ANN model for forecasting time series data. Applied Soft Computing*, 23, 27-38.

[3] Chen, S., Ge, L. (2019), *Exploring the attention mechanism in LSTM-based Hong Kong stock price movement prediction. Quantitative Finance*, 19, 1507-1515.

[4] Ellis, C.A., Parbery, S.A. (2005), *Is smarter better? a comparison of adaptive, and simple moving average trading strategies. Research in International Business and Finance*, 19(3), 399-411.

[5] Fischer, T., Krauss, C. (2018), *Deep learning with long short-term memory networks for financial market predictions. European Journal of Operational Research*, 270, 654-669.

[6] Géron, A. (2019), *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. Adaptive Computation and Machine Learning series. O'Reilly Media, Canada*.

[7] Goodfellow, I., Bengio, Y., Courville, A. (2016), *Deep Learning. Adaptive Computation and Machine Learning series. The MIT Press, Cambridge, Massachusetts*.

[8] Hassani, H., Dionisio, A., Ghodsi, M. (2010), *The effect of noise reduction in measuring the linear and nonlinear dependency of financial markets. Nonlinear Analysis, Real World Applications*, 11, 492-502.

[9] He, F., He, X. (2019), *A continuous differentiable wavelet shrinkage function for economic data denoising, Computational Economics*, 54, 729-761.

[10] Jung, G., Choi, S.-Y. (2021), *Forecasting foreign exchange volatility using deep learning autoencoder-LSTM techniques. Complexity*, (6647534).

[11] Moon, K.-S., Kim, H. (2019), *Performance of deep learning in prediction of stock market volatility. Economic Computation and Economic Cybernetics Studies and Research,* 53(2), 77-92.

[12] Moon, K.-S., Kim, H. (2023), *Efficient asset allocation based on prediction with adaptive data selection. Economic Computation and Economic Cybernetics Studies and Research,* 57(1), 57-72.

[13] Nabipour, M., Nayyeri, P., Jabani, H., Mosavi, A., Salwana, E., Shahab, S. (2020), *Deep learning for stock market prediction. Entropy*, 22(8), 840.

[14] Raudys, A., Lenciauskas, V., Malcius, E. (2013), *Moving averages for financial data smoothing. ICIST 2013, Information and Software Technologies*, 34-45.

[15] Sezer, O.B., Gudelek, M.U., Ozbayoglu, A.M. (2020), *Financial time series forecasting with deep learning: A systematic literature review: 2005-2019. Applied soft computing journal*, 90 (106181).

[16] Sherstinsky, A. (2020), *Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. Physica D*, 404, 132306.

[17] Song, D., Baek, A.M.C., Kim, N. (2021), *Forecasting stock market indices using padding-based Fourier transform denoising and time series deep learning models. IEEE Access,* 9, 83786-83796.

[18] Su, Z., Xie, H., Han, L. (2021), *Multi-factor RFG-LSTM algorithm for stock sequence predicting. Computational Economics*, 57, 1041-1058.

[19] Sun, E.W., Meinl, T. (2012), *A new wavelet-based denoising algorithm for high-frequency financial data mining. European Journal of Operational Research*, 217(3), 589-599.

[20] Wu, D., Wang, X., Wu, S. (2021), *A hybrid method based on extreme learning machine and wavelet transform denoising for stock prediction. Entropy*, 23(4), 440.