Hindawi
Scientific Programming
Volume 2022, Article ID 3798734, 13 pages
https://doi.org/10.1155/2022/3798734

*Research Article*

# An Online Kernel Adaptive Filtering-Based Approach for Mid-Price Prediction

**Shambhavi Mishra**,[1] **Tanveer Ahmed**,[1] **Vipul Mishra**,[1] **Sami Bourouis**,[2] **and Mohammad Aman Ullah**[3]

[1]*School of Engineering and Applied Sciences, Bennett University, Greater Noida 201310, India*
[2]*Department of Information Technology College of Computers and Information Technology, Taif University, Taif 21944, Saudi Arabia*
[3]*Department of Computer Science and Engineering, International Islamic University Chittagong, Chittagong, Bangladesh*

Correspondence should be addressed to Mohammad Aman Ullah; aman_cse@iiuc.ac.bd

Received 30 August 2021; Revised 25 October 2021; Accepted 27 November 2021; Published 15 February 2022

Academic Editor: Punit Gupta

The idea of multivariate and online stock price prediction via the kernel adaptive filtering (KAF) paradigm is proposed in this article. The prediction of stock prices is traditionally done with regression and classification, thereby requiring a large set of batch-oriented and independent training samples. This is problematic considering the nonstationary nature of a financial time series. In this research, we propose an online kernel adaptive filtering-based approach for stock price prediction to overcome this challenge. To examine a stock's performance and demonstrate the work's superiority, we use ten different KAF family of algorithms. In this paper, we take on this challenge and propose an approach for predicting stock prices. To analyze a stock's performance and demonstrate the work's superiority, we use ten distinct KAF algorithms. Besides, the results are analyzed on nine-time windows such as one day, sixty minutes, thirty minutes, twenty five minutes, twenty minutes, fifteen minutes, ten minutes, five minutes, and one minute. We are the first to experiment with several time windows for all fifty stocks on the Indian National Stock Exchange, to the best of our knowledge. It should be noted here that the experiments are performed on stocks making up the main index: Nifty-50. In terms of performance and compared to existing methods, we have a 66% probability of correctly predicting a stock's next upward or downward movement. This number clearly shows the edge that the proposed method has in actual deployment. Furthermore, the experimental findings show that KAF is not only a better option for predicting stock prices but that it may also be used as an alternative in high-frequency trading due to its low latency.

## 1. Introduction

Time-series prediction is prevalent in economics and investment research. Stock price prediction is one of the most popular applications of time-series prediction. Its success stems from its ability to reduce asset management costs, market impacts, and volatility risks [1]. It is a commonly held notion that stock markets are complex, volatile, and chaotic [2]. The markets, in our perspective, are made up of a variety of factors that influence stock movement. Predicting stock's value at any given time in the future is, therefore, an important problem of academia and industry. Previous studies [3] have shown that the prediction of stock prices, particularly with the nonstationary and the nonlinear nature of the underlying asset, is challenging. In this regard, several models have been proposed, but the problem is nowhere near its end [4], and a substantial improvement is required. In addition, studies have also extended the problem by predicting option prices, volatility [5], and so on. This significant body of work demonstrates that stock price prediction remains a significant issue requiring solutions to a wide range of problems.

As discussed in the previous paragraph, stock price prediction is a significant challenge. In this regard, a plethora of techniques have been used for predicting stock prices, such as neural network (NN), support vector machine (SVM), genetic algorithm, fuzzy logic, and Bayesian model [6]. However, getting an optimal solution is still a long way to go. During our literature review, we discovered that current research has overlooked kernel adaptive filtering (KAF) and has not thoroughly investigated this paradigm for financial time-series forecasting, especially stock prediction. Although there are a few introductory studies [7], a large-scale comprehensive evaluation lacks literature. With this shortcoming in mind, we would like to emphasize that KAF can be an effective stock prediction tool. The following observations serve as the foundation for our argument: first, KAF-based algorithms have a faster convergence rate; that is, the algorithm requires fewer iterations. Second, KAF has demonstrated excellent performance in nonstationary time-series prediction [8]. Third, KAF algorithms exhibit universal function approximation properties useful in highly dynamic environments [9]. Lastly, KAF has been used extensively in chaotic time-series prediction [10, 11]. Hence, it is also worth exploring the idea in financial time-series prediction. Therefore, in this article, we use the concept of KAF to examine and comprehend the real-time movement of stock prices. In addition to KAF being one of the largest unexplored paradigms in stock prediction, the literature review revealed one more issue. It contains one of the problems related to batch learning. We believe that sequential learning is the best tool rather than batch learning for financial time-series forecasting. This is mainly because a financial time series is nonstationary. We further argue that expecting a model trained on offline samples to perform excellently in a real situation is a slippery slope. The rationale here is supported by the work presented in the literature, which claims that online learning is the best way to understand and interpret nonstationary data behavior [12]. Consequently, studies have shown that online learning can be an effective method [13]. It is based on the concept of sequential measurement (training is performed sample-by-sample and in real time). Various scenarios can easily be added, and the algorithm adjusts the weight vector to provide accurate predictions. As a result, in order to solve the issue stated in the article, we enhance the KAF idea with online learning.

With respect to the challenges and the ideas discussed in this section, we present an online KAF algorithms to predict the price of stock. The use of KAF techniques to stock price prediction is still limited [7, 14]. However, the concept is based upon this study is precedent and builds upon it to extend the application of KAF to a broader range of environments and contexts. With data taken from Nifty-50, the Indian Stock Index, we first build our dataset consisting of prices collected at a time window of one day, sixty minutes, thirty minutes, twenty five minutes, twenty minutes, fifteen minutes, ten minutes, five minutes, and one minute. These windows are chosen as they are some of the most common windows looked at by day traders. It should be noted here that the prices are collected for a total of fifty companies (they make up the main index: Nifty-50). Subsequently, we apply the ideas on each of the time windows and predict the next potential number for the "mid-price" of the stock. With comprehensive numerical investigation, we have found that the proposed trading algorithm has an extra 16% edge in the field, thereby making it an effective method capable of generating good returns in the long run. The following are the paper's key contributions:

(1) A novel KAF-based online method for forecasting a stock's mid-price is introduced. We look at two situations in which the mid-price is measured as (high + low)/2 and (open + close)/2, respectively. The main motivation for looking into mid-price was that mid-price time series is less noisy than close-price time series.

(2) With a comprehensive investigation performed on nine different time windows. We discover the best window for predicting stock prices. In the literature, several authors have focused on predicting daily prices [15, 16]. We, however, show that focusing efforts on other time windows could also be optimal.

(3) In this article, ten different KAF algorithms are used, and a detailed analysis is presented to validate the work. To the best of our knowledge, an investigation of this magnitude eludes literature.

The following section has been divided into sections. The methods proposed by various researchers in the subject of stock prediction are discussed in Section 2. Proposed methodology is described in Section 3. The experiments performed with different KAF algorithm, and their results are included in Section 4. Finally, in Section 5, the conclusions and future scopes are described.

## 2. Related Work

The work of other authors in the field of stock prediction is discussed in this section. Predicting stock has remained one of the nontrivial issues of the literature [17]. Previous studies have shown that the prediction of stock prices is difficult due to the inherent nonstationary behavior in the data [18]. Several studies [5] have shown that stock prediction is challenging and noisy. Various linear techniques such as correlations, discriminating analysis, autoregressive models, and moving averages have also been studied in the past [19]. Machine learning (ML) has been a popular field in time-series prediction in recent years. ML-based techniques are explored heavily as they can recognize complex patterns in stock prices [20]. Due to the nonlinear and time-varying nature of time-series, there has recently been a surge in demand for online prediction algorithms [21]. Online algorithms use the sequential calculation to achieve reliable and faster outcomes [13]. In this regard, several techniques have been developed, such as online support vector regression (SVR), NN [8], and KAF algorithms [10]. NN methods take a lot of processing power and have a slow convergence rate [22]. SVR provides superior applicability; however, it is not appropriate for huge datasets. Furthermore, the multifilter neural network (MFNN) is investigated, and it is discovered that MFNN outperforms SVR, random forests, and other neural network-based approaches. The use of convolutional neural networks (CNN) has also been explored to predict the next-day prices [23]. CNN outperformed for multimodality images in the biomedical domain [24, 25]. Furthermore, for stock price prediction, long short-term memory (LSTM) is applied [26]. The authors used an LSTM network with a single layer and 200 nodes in [27]. Furthermore, the network employs a single-layer LSTM with 140 nodes [28]. In contrast to using a deep architecture with four LSTM layers and 96 nodes in the hidden layers, each LSTM layer was further followed by a dropout layer [29].

Adaptive filtering has been proven to be a preferable choice for streaming data having nonstreaming behavior [11, 30, 31]. For sequential stock prediction, KAF can be used by exploiting market interdependence. Fast convergence, low computational complexity, and nonparametric behavior make KAF a preferable choice [10, 32]. One research [33] focuses on adaptive asynchronous differential evolution with trigonometric mutation modified mutation operation, and adaptive parameters modified the convergence speed and diversity. In [34], the authors proposed meta-cognitive recurrent kernel online learning for multistep predictions of stocks. Although these studies show the potential that KAF has, KAF has not been investigated thoroughly in the context of stock price prediction. Though there are few studies in literature focusing on the area, large-scale investigation eludes literature. Nevertheless, we must point that the work presented in [14] proposes a two-phase method for stock prediction. First, sequential learning using KAF was applied to learn the underlying model for each stock separately. In the second phase, to improve prediction, real time models are learned from different stock. In [7], the authors proposed the idea of multikernel adaptive filters for online options trading. The method was applied to Taiwan composite stock index. Garcia-Vega et al. [35] presented a multikernel learning approach to overcome the two primary concerns with KAF: kernel size and step size. Despite the fact that these papers concentrate on using the KAF paradigm to forecast stock prices, none of them validates the paradigm's effectiveness on a large-scale dataset. Moreover, the impact of multiple time windows is not considered. In our opinion, testing the method on multiple time windows that are often looked at by traders is of prime importance.

## 3. Methodology

*3.1. Brief Discussion on KAF.* We work with online learning-based KAF techniques, as discussed in Section 1. The purpose of KAF is to learn with well-known input-output mapping $f\colon S \longrightarrow R$, and it contains sequence of data such as $((s_1, d_1), (s_2, d_2), \ldots, (s_i, d_i))$, where, $S \subseteq R^L$ is the input space, $s_i$, $i = 1, \ldots, n$, is the system input at sample time, and $d_i$ is known as desired response. In reproducing kernel Hilbert space (RKHS) $F$, KAF transforms the data into a set of points. Inner products can then be used to solve the problem. There is no need to do expensive computations in high-dimensional space, owing to the famous "kernel trick." In KAFs, generally, the computation involves the use of a kernel. The following equation is an example of a kernel:

$$\kappa < s, s' > \; = \exp \frac{\left(\left\| s - s' \right\|^2\right)}{\sigma^2}, \tag{1}$$

where $\sigma$ represents the kernel width.

*3.2. Kernel Adaptive Filtering Algorithms.* In this subsection, we briefly describe the ten different KAF methods.

*3.2.1. Least Mean Square (LMS).* The LMS algorithm, according to [36] employs a finite impulse response (FIR) filter, also known as a traversal filter, whose output is based on a linear combination of the input presented in the following equation:

$$y_i = \varpi_{(i-1)}^T s_i, \tag{2}$$

where $\varpi_{(i-1)}$ represents the weight vector at iteration $(i-1)$. The following equation contains the main idea of the LMS algorithm:

$$\begin{aligned} \varpi_0 &= 0, \\ e_i &= t_i - \varpi_{(i-1)} s_i, \\ \varpi_i &= \varpi_{(i-1)} + \eta e_i s_i, \end{aligned} \tag{3}$$

where $\eta$ and $e_i$ stands for step size and prior error. The weight-update equation findings were represented in the following equation:

$$\varpi_i = \eta \sum_{i=1}^{N} e_i s_i, \tag{4}$$

The following equation represents the inner product:

$$\begin{aligned} t = \varpi_i(s) &= \eta \sum_{i=1}^{n} e_i <s_i, s>, \\ e_i &= t_i - \eta \sum_{i=1}^{n-1} e_i <s_i, s>. \end{aligned} \tag{5}$$

*3.2.2. Kernel Least Mean Square (KLMS).* To derive KLMS [36], the input $(s_i)$ is converted into $F$ as $\phi(s_i)$. Using LMS, we can now rewrite the input and output mapping as follows:

$$\begin{aligned} \varpi_0 &= 0, \\ e_i &= d_i - \varpi_{(i-1)}^T \phi(i), \\ \varpi_i &= \varpi_{(i-1)} + \eta e_i \phi(i), \end{aligned} \tag{6}$$

where $e_i$ is represented as the prediction error, $\eta$ is the size of every step, and $\phi(s_i)$ is defined as the transformed filter input at a certain point in time or iteration $i$. Equation (7) compute the result, where we can use the famous kernel tricks. Consequently, the model now becomes

$$\begin{aligned} f_0 &= 0, \\ e_i &= d_i - f_{i-1}(s_i), \\ f_i &= f_{i-1} + \eta e_i \kappa <s_i, .>. \end{aligned} \tag{7}$$

In KLMS, a new unit of the kernel is assigned to all new samples points with $\eta e_i$ as the coefficient value. Following the radial basis function (RBF) described in this section, the system is represented as follows:

$$f_i = \sum_{j=1}^{i} o_j(i)\kappa <s_j, .>. \tag{8}$$

The coefficients $o(i)$ and the centers $C(i) = \{s(j)\}_{j=1}^{i}$ are saved inside the storage during the training process.

*3.2.3. Kernel Affine Projection Algorithm (KAPA).* KAPA [37] is used where we want to improve the performance owing to the gradient noise. In KAPA, we estimate using the weight vector $\varpi$ and minimise the cost function with the sequences $\{d_1, d_2\}$ and $\{\phi(1), \phi(2)\}$ as shown below

$$\min_{\varpi \text{emp}} \left| d - \varpi^T \phi(s) \right|^2. \tag{9}$$

We replace the concept of covariance and cross variance matrix-vector by local approximation directly from the data using stochastic gradient descent summarized in

$$\varpi_i = \varpi_{(i-1)} + \eta\psi(i)\left[d(i) - \psi(i)^T \varpi_{(i-1)}\right], \tag{10}$$

where $\psi(i) = [\phi(i-K+1), \ldots, \phi(i)]$ and K is the observation and regressor.

*3.2.4. Leaky Kernel Affine Projection Algorithm (LKAPA).* LKAPA [37] is the extension of KAPA as discussed in Section 3.2.3.Based on the selected kernels, the feature space can be infinitely dimensional, where the weight updation task is difficult. In the common consideration, the solution is the modification in equation (10) as follows.

The weight vector in Equation (11) is calculated using the following criteria:

$$\varpi_i = \sum_{j=1}^{i} o_j(i)\phi(i), \quad \forall_i \geq 0. \tag{11}$$

Equation (12) is used to reduce the following objective function from the perspective of empirical risk minimization:

$$\min_{\varpi \text{emp}} \left| d - \varpi^T \phi(s) \right|^2 + \Lambda \|\varpi\|^2. \tag{12}$$

Then, we get the updated weight, and it is shown in

$$\varpi_i = (1 - \Lambda\eta)\varpi_{(i-1)} + \eta\psi(i)\left[d(i) - \psi(i)^T \varpi_{(i-1)}\right], \tag{13}$$

where $\psi(i) = [\phi(i-K+1), \ldots, \phi(i)]$.

Finally, coefficient $o_\kappa(i)$ is updated as

$$o_\kappa(i) = \begin{cases} k = i, \eta\left(d_i - \sum_{j=1}^{i-1} o_j(i-1)k_{i,j}\right) \\ \text{for}\{i-K+1 \leq k \leq i-1\}(1-\Lambda\eta)o_k(i-1) + \eta\left(d(k) - \sum_{j=1}^{i-1} o_j(i-1)\kappa_{k,j}\right). \\ 1 \leq k < i-K+1 (1-\Lambda\eta)o_k(i-1) \end{cases} \tag{14}$$

*3.2.5. Normalized Online Regularized Risk Minimization Algorithm (NORMA).* Similarly, the LKAPA [37] extension comes in NORMA, and also it is related to KAPA discussed in Section 3.2.3. It also includes the regularization and nonfunctional approaches.

*3.2.6. Quantized Kernel Least Mean Square Algorithm (QKLMS).* Quantization techniques are used in various applications such as digitization, data compression, speech, and image coding. QKLMS is a famous algorithm proposed in [11], which deals with the issue of data redundancy. The computational complexity of QKLMS and KLMS is nearly identical. The main difference between the two algorithms is that QKLMS uses redundant data to update the coefficient of closest centre in real time. The following equation represents the main idea using the quantization operator:

$$\begin{aligned} \varpi_0 &= 0, \\ e_i &= t_i - \varpi_{(i-1)}^T \phi(i), \\ \varpi_i &= \varpi_{(i-1)} + \eta e_i \mathcal{Q}[\phi(i)], \end{aligned} \tag{15}$$

where $\mathcal{Q}[.]$ signifies the quantization in feature space $F$. The following equation summarises the learning rule for QKLMS:

$$\begin{aligned} f_0 &= 0, \\ e_i &= t_i - f_{i-1}(s_i), \\ f_i &= f_{i-1} + \eta e_i \kappa(Q[s_i]). \end{aligned} \tag{16}$$

*3.2.7. Fixed Budget Quantized Kernel Least Mean Square Algorithm (FBQKLMS).* The FBQKLMS [38] deals with the increasing popularity of online kernel approaches. The suggested algorithm uses a significance measure-based pruning criterion based on the weighted contribution of existing data centres.

*3.2.8. Kernel Adaptive Filtering with Maximum Correntropy Criterion (KMCC).* The fundamental goal of the method is to maximise the crossentropy between the desired $d_i$ and actual output $y_i$ [39]. Using the MCC technique [39] and SGD, the algorithm can be written as follows:

$$\begin{aligned}
&\varpi_0 = 0,\\
&\varpi_{(i+1)} = \varpi_i + \eta\frac{\partial\kappa_\sigma\left(t_i, \varpi_i^T\phi(s_i)\right)}{\partial\varpi_i}; = \varpi_i + \eta\left[\left(\exp\frac{\left(-e_i^2\right)}{2\sigma^2}\right)e_i\phi(i)\right],\\
&\ldots = \eta\sum_{i=1}^{n}\left[\left(\exp\frac{\left(-e_i^2\right)}{2\sigma^2}\right)e_i\phi(i)\right],\\
&y_i = \eta\sum_{i=1}^{n}\left[\left(\exp\frac{\left(-e_i^2\right)}{2\sigma^2}\right)e_i\kappa < s_i, s_n >\right], e_i = d_i - y_i,
\end{aligned} \tag{17}$$

where $\eta$ is the step size and $\sigma$ is the kernel width. The entire amount of error and prediction calculation can be summarized in equation (17).

*3.2.9. Multikernel Normalized Least Mean Square (MKNLMS).* According to [30], the KNLMS algorithm is used to create dictionaries based on the coherence requirement. Here, we explore at KNLMS through the perspective of MKNLMS-CS (multi-kernel normalised least mean square algorithm with coherence-based sparsification). Consider the empty dictionary at the initial stage represented as ($\mathcal{J}_0^{cs} := \varnothing$), by which the $H_0$ is shown as an empty matrix as $M^*$. Consider the Hilbertian unit for simplification of $\kappa(s, s) = 1, \forall_s \in s$, which is satisfied by the Gaussian kernel. $n$ is added into $\mathcal{J}_n^{cs}$ in the case when the defined condition holds in the proposed methodology presented in equation (18)

$$\|\kappa\|_{\max} := \max_{m\in\mathcal{M}}\max_{j\in\mathcal{J}_n^{cs}}\left|\kappa_m\left(s_n, s_j\right)\right|, \leq \phi, n \in N, \tag{18}$$

where $\eta \in [0, 2]$ and $\Lambda > 0$ denotes the step size and regularization parameter, respectively. $\delta > 0$ is the threshold.

Considering

If equation (18) is satisfied, $\mathcal{J}_{n+1}^{cs} := \mathcal{J}_n^{cs} \cup \{n\}$. If equation (18) is not satisfied, $j_{n+1}^{cs} := j_n^{cs} \cup \{n\}$:

$$\begin{aligned}
H_{n+1} &:= \overline{H}_n + \eta\frac{t_n - <\overline{K}_n, \overline{H}_n>}{\left\|\overline{K}_n\right\|^2 + \Lambda}\overline{K}_n,\\
H_{n+1} &:= H_n + \eta\frac{t_n - <H_n, K_n>}{\left\|K_n\right\|^2 + \Lambda}K_n,
\end{aligned} \tag{19}$$

where $\overline{H}_n := [H_n 0]$ and $\overline{K}_n := [K_n\overline{k}_n]$ with $\overline{k}_n := [\kappa_1\ (s_n, s_n), \kappa_2(s_n, s_n), \kappa_3(s_n, s_n), \ldots \kappa_M(s_n, s_n)]^T$ where $0 \in\in R^M$ is the zero vector. The value of $M$ for KNLMS is 1.

*3.2.10. Probabilistic Least-Mean Square Filter (PROB-LMS).* PROB-LMS [31] gives adaptable step-size to the LMS algorithm in Section 3.2.1. and also applied in the stationary and nonstationary environment. The LMS filter can be approximated effectively using a probabilistic approach. It includes a step-size LMS algorithm that may be modified as well as a measure of estimation uncertainty. It also maintains the standard LMS's linear complexity.

*3.3. Problem Formulation.* Our main objective, is to predict the stock's mid-price as stated in Section 1. The motive of stock price prediction is to calculate stock's future values depending on historical values. For this, we measured the percentage change in mid-price. As a result, we used the concept of order n auto-regression to predict future stock price changes. The sample regression equation is shown in Table 1. Multivariate financial time-series estimation often employs this formulation [40, 41] to predict future values of a time series. The formulation shown in Table 1 is done by considering daywise mid-prices. It should be noted here that the same procedure was used for all time windows. As a result, the problem was rephrased as follows: autoregression-based next percentage prediction. The exact mid-price of the stock may therefore be easily calculated using the percentage change. Figure 1 depicts the proposed approach's overall methodology. The Nifty-50 dataset was used in the experiments. We consider two different aspects for the mid-price prediction: (i) (high + low)/2 and (ii) (open + close)/2. As a result of this calculation, we created the dataset and preprocessed it using nine prediction windows (one-minute, five-minutes, ten-minutes, fifteen-minutes, twenty-minutes, twenty-five minutes, thirty-minutes, sixty-minutes, and one day). Further, the percentage change was calculated for each time windows, and min-max normalization was applied. The selection of embedding dimension $(M)$ is a difficult task. We choose different $M \in \{2, 3, 4, 5, 6, 7\}$ and set the maximum dictionary size for required algorithms to 500 with Gaussian

Table 1: A one-day time frame (Stock-TITAN).

| Day | High price | Low price | (high + low)/2 | Change in price |
|---|---|---|---|---|
| 1 day | 1573 | 1555.95 | 1564.475 | −0.765 4 |
| 2 days | 1567 | 1538 | 1552.5 | 0.430 0 |
| 3 days | 1576.85 | 1541.5 | 1559.175 | 2.079 6 |
| 4 days | 1621.35 | 1561.85 | 1591.6 | −2.456 6 |
| 5 days | 1570 | 1535 | 1552.5 | −0.908 2 |
| 6 days | 155.3 | 1521.5 | 1538.4 | NA |

If we choose $M = 3$, then **Input** = [{−0.765 4, 0.430 0, 2.079 6}.], **Output** = [{−2.456 6}].

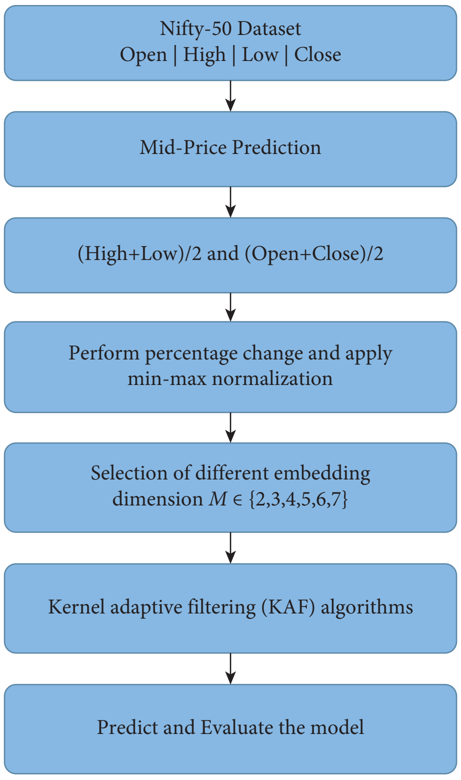
Nifty-50 Dataset
Open | High | Low | Close
Mid-Price Prediction
(High+Low)/2 and (Open+Close)/2
Perform percentage change and apply min-max normalization
Selection of different embedding dimension $M \in \{2,3,4,5,6,7\}$
Kernel adaptive filtering (KAF) algorithms
Predict and Evaluate the model

Figure 1: Proposed mid-price prediction framework.

kernel for each time window. In Table 1, we have shown an example considering the time window of 1 day (Stock-TITAN). The error estimation was performed with the help of ten different KAF algorithms for each time window. For example, the performance of each algorithm is analyzed to find which embedding dimension produces the best result. After getting the best embedding dimension for each algorithm, the embedding dimension that produces the best result and the corresponding algorithm is selected.

## 4. Experiments and Results

*4.1. Dataset Description.* This section explores into the specifics of the dataset that was used to test the applicability of the proposed method. For this study, we used data from the National Stock Exchange of India. The main index of NSE, Nifty-50, has 50 stocks. Based on the average and total daily turnover for equity shares, Nifty-50 is India's largest stock exchange. We collected data between January 01, 2021, and May 31, 2021, from 9:15 a.m. to 3:30 p.m. In addition, experimentation data are available at https://shorturl.at/lnvF2. The original data included open, high, low, and close (OHLC) prices and were available for one minute. The original data consisted of OHLC prices and were available for one minute. The dataset is generated and preprocessed in accordance with the nine prediction windows (one minute, five minutes, ten minutes, fifteen minutes, twenty minutes, twenty five minutes, thirty minutes, sixty minutes, and one day). Data samples range is different according to their time window. As pointed out in Section 1, we are trying to predict the mid-price with two different scenarios: (high + low)/2 and (open + close)/2. For this, firstly we calculated the percentage change of mid-price. All data values were normalized between zero to one range. Ten different KAF algorithms were used to the final preprocess data and each stock and analyse the comparative performance.

*4.2. Evaluation Criterion.* To measure and analyse the efficacy of various KAF algorithms, standard assessment criteria are used. In Table 2, $y_i$ and $d_i$ represent the actual and predicted output. $n$ is the time step, and

$$D_i = \begin{cases} 0, & \text{otherwise}, \\ 1, & (y_i - y_{i-1})(d_i - d_{i-1}) \geq 0, \end{cases} \tag{20}$$

Calculating the evaluation metrics with Nifty-50.

(1) The parameter listed in Table 3 were tuned manually. The parameter description for ten different algorithms are presented in Table 3. These values were found after multiple rounds of experimentation.

(2) For the error values, we applied the methods to all stocks and tried to quantify the predictive performance via the metrics discussed in this section. In total, we get $50 \times 3$ (one for each stock) error values for MSE, MAE, and DS, respectively.

(3) Then, for each of the 50 stocks, error estimation was performed using nine different prediction windows for ten different KAF algorithms.

(4) Finally, we used the average of all fifty-error metrics for a single time window and a single stock to reach the final value, which is presented in Tables 4 and 5. On all 50 stocks, the provided number represents the models' overall predictive capacity.

*4.3. Prediction, Convergence, and Residual Analysis.* In this subsection, we examined prediction, converge and residual analysis with the help of KAF algorithms. Regarding this, we have shown the prediction graphs with the KAPA algorithm (discussed in Section 3.2.3) for one stock (TITAN). Figure 2 shows the results for (high + low)/2, while Figure 3 shows the results for (open + close)/2. The predictive curve suits well against the original curve, as can be seen from the prediction graphs. It is worth noting that we have only given results for one prediction window (thirty-minutes) with one stock (TITAN). However, we must note that other stocks in the dataset produced similar result. The prediction graphs clearly show that the predictions are not exact, although they are close. To be precise, the numbers for MSE and MAE are presented in Tables 4 and 5. We must point out that getting accurate value in financial time series forecasting is tough. The goal has always been to get close enough values. Therefore, the result that we achieved shows the good predictive capability of the work. Figures 4 and 5 show the convergence graph for mid-price for (high+low)/2, (open+close)/2, respectively. We have provided the results using the KAPA algorithm with only one prediction window (thirty minutes) and one stock (TITAN), similar to the prior scenario. The algorithm converges quickly, as evidenced by the graphs, at the 1000th data point. We can see in KAF algorithms capacity to adapt and converge quickly. One more important point to note from the convergence graphs is that although there is some fluctuation in the graphs, it is nevertheless acceptable. This is because there will be noise in the new data and minor changes are inevitable. In addition to the results discussed so far, we have complemented the analysis by presenting the distribution of error residuals in Figures 6 and 7. It can be seen from the figures that residuals follow a normal distribution. Moreover, the outliers are also less. Furthermore, the residual's variance is low, demonstrating the KAF algorithm's superior prediction capability and potential in predicting the next immediate, mid-priced occurrence. Directional symmetry is used to determine the continuity of actual and expected prices in terms of stock movement. It is a measure for determining a model's ability to predict a stock's direction. We examined the ten different algorithms mentioned in Section 3 to better understand the actions of a stock's movement. The experiment revealed that using KNLMS, we have a 66% percent chance of accurately predicting the next up or down movement. This is shown in Table 5. The best result is obtained at the window of ten minutes, and the worst result is obtained at the one-minute window. From the table, it is also visible that there is a big difference in the number obtained for the one minute window and that for the rest of the windows. This is expected as there is much noise in a minute, which indeed affects prediction. It should be noted here that literature often ignores looking at these different time windows. Work mostly focuses on predicting daily prices [42, 43]. We discovered the perfect balance by playing with various time windows. Furthermore, when trading, it is recommended to strike a balance between error minimization and directional symmetry.

*4.4. Comparative Evaluation of KAF Algorithms.* Since we have used ten algorithms in our experimentation, it becomes essential to compare their performance. In this context, we present the topic in two separate situations. First, we analyze the results considering mid-price as (high + low)/2 to find the best algorithm. In the next scenario, we tried mid-price using (open + close)/2. Tables 4 and 5 show the outcome of this experiment. In terms of MSE and MAE, the tables show that KAPA outperforms other algorithms. When it comes to directional symmetry, we can see a contradiction. In directional symmetry, we see a conflict. Here, NORMA and KNLMS give the best performance.

*4.5. Comparison with Methods of a Similar Kind.* We have also compared the result with other existing techniques such as [28, 29] and [44]. These are some of the most recent deep learning (DL)-based algorithms for predicting stock prices. It should be noted here that these methods were trained and tested using 80:20 splits for 25 epochs. The time taken to train and make prediction was recorded. Specifically, these methods [28, 29] and [44] were reimplemented based on the architecture details and hyper-parameters setting found in the respective papers. The Nifty-50 dataset was used to train all of the methods. To ensure consistency across different methods for experimentation, we use sixty-minute time periods, for fifty stocks. All of the methods' results were then compared to the proposed method. Table 6 contains the

Table 2: Evaluation metrics.

| MSE | MAE | DS |
|---|---|---|
| $\sum_{i=1}^{n} (y_i - d_i)^2$ | $(1/n) \sum_{i=1}^{n} \lvert y_i - d_i \rvert$ | $(1/n) \sum_{i=1}^{n} D_i$ |

Table 3: Parameter description of KAF techniques for NSE-50 dataset for mid-price.

| Parameter | KAPA | KLMS | KMCC | KNLMS | FBQKLMS | LKAPA | LMS | NORMA | PROB-LMS | QKLMS |
|---|---|---|---|---|---|---|---|---|---|---|
| ($\sigma$) | 5.0 | 7.0 | 4.0 | 7.0 | 5 | 6 | — | 3 | — | 4 |
| ($\eta$) | 1.5 | 1.7 | 1.7 | 1.7 | 0.2 | 0.03 | — | — | — | 0.2 |
| ($\epsilon$) | 1E-4 | — | — | 1E-4 | 0.4 | — | — | 1.5 | — | 0.5 |
| ($\beta$) | — | — | — | — | 0.85 | — | — | — | — | — |
| ($\Lambda$) | — | — | — | — | — | 1E-2 | — | 1E-2 | 0.4 | — |
| ($\sigma_2 n$) | — | — | — | — | — | — | — | — | 2 | — |
| ($\sigma_2 d$) | — | — | — | — | — | — | — | — | 3 | — |
| mu0 | 0.2 | — | — | 2 | — | — | 0.2 | — | — | — |
| (P) | 20 | — | — | — | — | 20 | — | — | — | — |
| $\tau$ | — | — | — | — | — | — | — | 5000 | — | — |
| tcoff | — | — | — | — | — | — | — | 4 | — | — |

$\sigma$ = kernel width, $\sigma_2 n$ = variance of observation noise, $\sigma_2 d$ = variance of filter weight diffusion, $\eta$ = step-size, $\epsilon$ = regularization parameter, $\Lambda$ = Tikhonov regularization, tcoff = learning rate coefficient, $\tau$ = memory size (terms retained in truncation), mu0 = coherence criterion threshold, P = memory length, nu = approximate linear dependency (ALD) threshold, and $\beta$ = forgetting factor for influence.

Table 4: Result in terms of MSE, MAE, and DS for mid-price (high + low)/2.

| Time window | MSE | Best algorithms out of ten discussed (according to MSE) | MAE | Best algorithms out of ten discussed (according to MAE) | DS | Best algorithms out of ten discussed (according to DS) |
|---|---|---|---|---|---|---|
| 1 day | 0.030 6 | KAPA | 1.412 9 | KAPA | 0.537 8 | NORMA |
| 60 minutes | 0.009 1 | KAPA | 0.509 6 | KAPA | 0.559 2 | NORMA |
| 30 minutes | 0.005 3 | KAPA | 0.359 5 | KAPA | 0.555 8 | PROB-LMS |
| 25 minutes | 0.004 7 | KAPA | 0.331 4 | KAPA | 0.557 8 | NORMA |
| 20 minutes | 0.003 8 | KAPA | 0.290 9 | KAPA | 0.555 0 | NORMA |
| 15 minutes | 0.003 0 | KAPA | 0.253 4 | KAPA | 0.555 6 | NORMA |
| 10 minutes | 0.002 1 | KAPA | 0.201 9 | KAPA | 0.547 2 | NORMA |
| 5 minutes | 0.001 2 | KAPA | 0.144 7 | KAPA | 0.534 2 | PROB-LMS |
| 1 minute | 0.000 27 | KAPA | 0.059 0 | KAPA | 0.549 6 | NORMA |

Table 5: Result in terms of MSE, MAE, and DS for mid-price (open + close)/2.

| Time window | MSE | Best algorithms out of ten discussed (according to MSE) | MAE | Best algorithms out of ten discussed (according to MAE) | DS | Best algorithms out of ten discussed (according to DS) |
|---|---|---|---|---|---|---|
| 1 day | 0.032 4 | KAPA | 1.298 9 | KAPA | 0.597 0 | NORMA |
| 60 minutes | 0.009 0 | KAPA | 0.454 1 | KAPA | 0.636 7 | NORMA |
| 30 minutes | 0.004 8 | KAPA | 0.316 8 | KAPA | 0.654 7 | NORMA |
| 25 minutes | 0.004 1 | KAPA | 0.290 3 | KAPA | 0.659 2 | NORMA |
| 20 minutes | 0.003 3 | KAPA | 0.254 3 | KAPA | 0.658 0 | NORMA |
| 15 minutes | 0.002 6 | KAPA | 0.219 8 | KAPA | 0.665 2 | KNLMS |
| 10 minutes | 0.001 7 | KAPA | 0.174 7 | KAPA | 0.666 6 | KNLMS |
| 5 minutes | 0.000 9 | KAPA | 0.122 4 | KAPA | 0.662 8 | KNLMS |
| 1 minute | 0.000 25 | KAPA | 0.055 2 | KAPA | 0.601 8 | KNLMS |

results. The table's data clearly demonstrate the proposed method's superiority over a number of other ways

*4.6. Effect of Different Kernels.* In Table 7, we can see the effect of different kernel methods. For this test, we used a thirty-minute time window (Stock-TITAN) with the algorithm KAPA and analyzed the best performance of RBF kernel in terms of MSE. That is why, we have chosen RBF kernel (gauss) for each algorithm.

*4.7. Experimentation with Dictionary Size.* We also conducted experiments with various dictionary sizes. The result for this test is shown in Table 8. For this test, we used a thirty-minute time window with the algorithm KMCC. It is visible

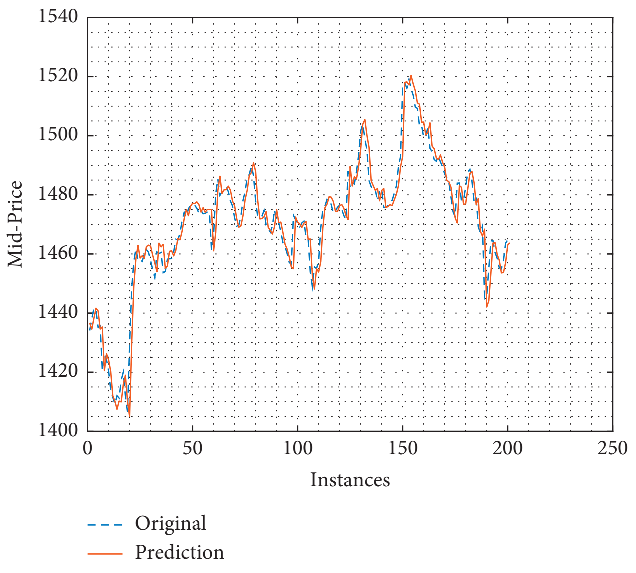

Figure 2: Prediction for one stock (TITAN) using KAPA (high + low)/2.

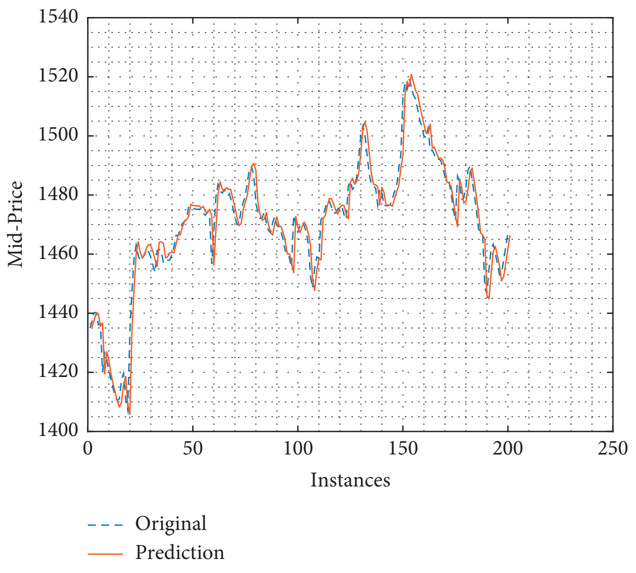

Figure 3: Prediction for one stock (TITAN) using KAPA (open + close)/2.

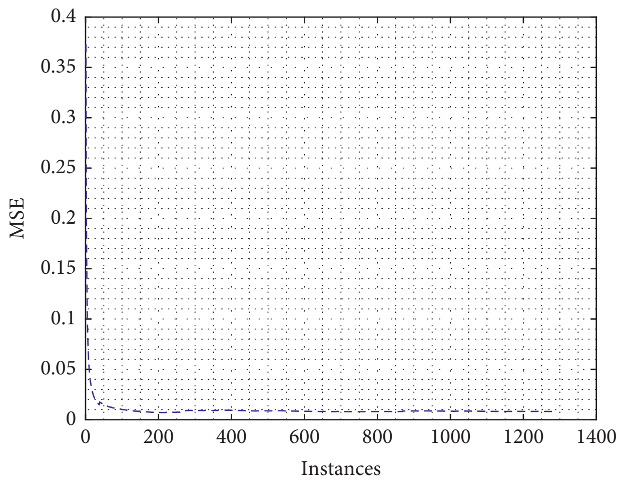

Figure 4: Error convergence for one stock (TITAN) using KAPA (high + low)/2.

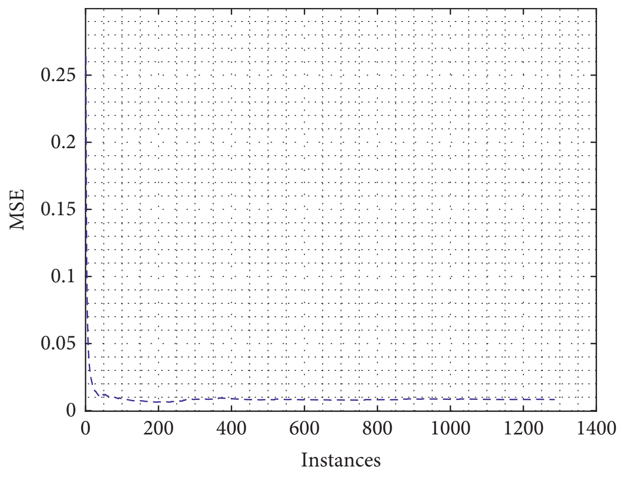

Figure 5: Error convergence for one stock (TITAN) using KAPA (open + close)/2.

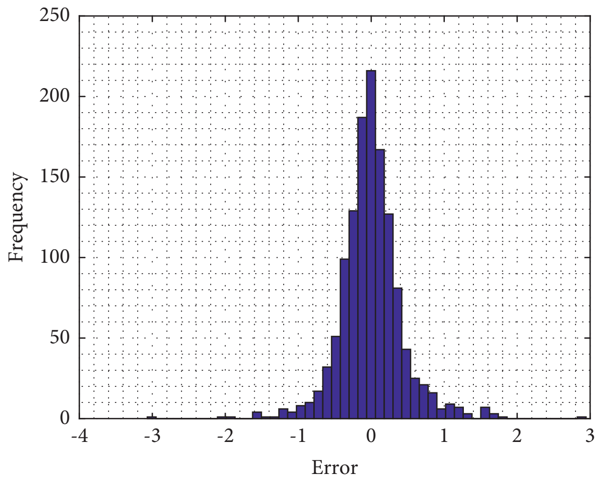

Figure 6: Error residuals for one stock (TITAN) using KAPA (high + low)/2.

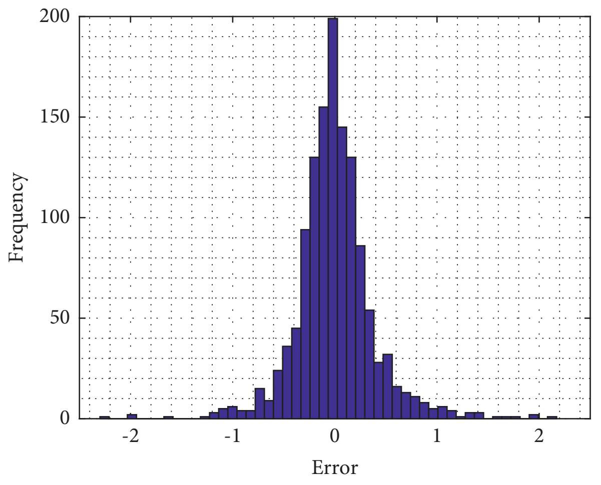

Figure 7: Error residuals for one stock (TITAN) using KAPA (open + close)/2.

Table 6: The proposed research is compared to different state-of-the-art stock prediction approaches.

| Method | MSE | Execution time (s) |
|---|---|---|
| LSTM [28] | 0.683 1 | 256.16 |
| LSTM [29] | 0.681 7 | 945.09 |
| LSTM [44] | 0.682 4 | 1770.40 |
| KAPA (proposed method) | **0.009 1** | 2.132 |

Table 7: Effect of different kernel methods on time window of thirty minutes (Stock-TITAN) using KAPA for (high + low)/2.

| | (High + low)/2 | | | (Open + close)/2 | | |
|---|---|---|---|---|---|---|
| Kernel function | MSE | MAE | DS | MSE | MAE | DS |
| RBF kernel (Gauss) | 0.008 120 | 0.298 57 | 0.534 21 | 0.008 27 | 0.260 47 | 0.531 88 |
| Anisotropic RBF kernel (Gauss-anis) | 0.019 19 | 0.406 17 | 0.515 55 | 0.025 35 | 0.348 68 | 0.579 31 |
| Laplace kernel (Laplace) | 0.008 128 | 0.298 72 | 0.533 43 | 0.008 28 | 0.260 61 | 0.531 10 |

Table 8: The influence of dictionary size.

| Dictionary size | MSE | MAE | DS | Execution time (s) |
|---|---|---|---|---|
| 500 | 0.014 8 | 0.368 | 0.545 8 | 0.820 2 |
| 1000 | 0.032 3 | 0.530 2 | 0.614 3 | 0.853 3 |
| 5000 | 0.011 37 | 0.308 0 | 0.644 6 | 1.314 |

The algorithm chose KMCC (30 minutes).

from Table 8 that increasing the dictionary size leads to an improvement in the system's performance. It should be noted here that when the size is 1000, the performance has fallen. The reason for this behavior could be the erratic behavior of the stock, the presence of noise, or too much irrelevant data. The exact reason is unknown. However, it is worth noting that with a dictionary size of 500, for forecasting a single stock, execution time is 0.82 seconds. This low number clearly shows the advantage one can achieve in high-frequency trading.

*4.8. Important Note: Error Minimization and Profitability.* We obtain an MSE of $10^{-4}$ as the lowest error. We can observe from Tables 4 and 5 that KAPA gives best results in terms of MSE and MAE. It is important to note that, in the one-minute time window, we reached a minimum error value. From Tables 4 and 5, we can also see that going down the column (for MSE and MAE only), the results are improving with the one-minute time window giving the best figures. However, because the time window is one minute, the volatility is low enough that decreasing error will not result in too much benefit. Moreover, there is too much noise while trading at one-minute window. To look at it another way, one-minute volatility is lower, resulting in very close predictions. However, in a low-volatility environment, the chances of taking a position and making a highly profitable trade are also low.

## 5. Conclusion and Future Work

This paper focuses on predicting a stock's mid-price. Predicting a financial nonstationary time series is an open fundamental and a nontrivial problem of literature. To address this, we proposed a framework based on online learning-driven KAF algorithms. In the proposed work, ten different KAF algorithms were evaluated and analyzed on Indian National Stock Exchange (Nifty-50). In contrast to the existing methods, experiments are performed on nine different time windows. This was done keeping in mind the method's applicability in intraday and swing trading. Previous studies often underestimated the importance of intraday time windows. We, therefore, tried to bridge this gap through the work presented here. The experimental results show the superiority and predictive capabilities of the work. The KAF class of algorithms was also discovered to be not only efficiently working in execution time but also providing best results of error minimization, demonstrating their importance in high-frequency trading. The goal of the research was to propose a KAF-based method for the prediction of stock's mid-price. The empirical results on Nifty-50 dataset show that the proposed method achieved superior performance over existing stock prediction methods. On voting schema KAPA shown better prediction performance with all-time windows, NORMA & KNLMS gave the best performance in terms of directional symmetry. It is worth noting that every KAF-based algorithm is hyperparameter-sensitive. As a result, in the future, we will experiment with various hyper-parameter optimization approaches in order to enhance the framework's predictive capabilities.

## Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

## Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the study.

## Acknowledgments

The authors would like to thank Taif University Researchers Supporting Project, Taif University, Taif, Saudi Arabia (no. TURSP-2020/26).

## References

[1] Y. S. Abu-Mostafa and A. F. Atiya, “Introduction to financial forecasting,” *Applied Intelligence*, vol. 6, no. 3, pp. 205–213, 1996.

[2] A. Abraham, N. S. Philip, and P. Saratchandran, “Modeling chaotic behavior of stock indices using intelligent paradigms,” 2004, https://arxiv.org/ftp/cs/papers/0405/0405018.pdf.

[3] M. P. Clements, P. H. Franses, and N. R. Swanson, “Forecasting economic and financial time-series with non-linear models,” *International Journal of Forecasting*, vol. 20, no. 2, pp. 169–183, 2004.

[4] M. Kumar and M. Thenmozhi, “Forecasting stock index movement: a comparison of support vector machines and random forest,” in *Proceedings of the Indian institute of capital markets 9th capital markets conference paper*, Mumbai, India, January 2006.

[5] R. Singh and S. Srivastava, “Stock prediction using deep learning,” *MultimedForecasting stock index movement: a comparison of support vector machines and random forestdia Tools and Applications*, vol. 76, no. 18, pp. 18569–18584, 2017.

[6] M. Vijh, D. Chandola, V. A. Tikkiwal, and A. Kumar, “Stock closing price prediction using machine learning techniques,” *Procedia Computer Science*, vol. 167, pp. 599–606, 2020.

[7] S.-C. Huang, C.-C. Chiou, J.-T. Chiang, and C.-F. Wu, “A novel intelligent option price forecasting and trading system by multiple kernel adaptive filters,” *Journal of Computational and Applied Mathematics*, vol. 369, p. 112560, 2020.

[8] M. Han, S. Zhang, M. Xu, T. Qiu, and N. Wang, “Multivariate chaotic time series online prediction based on improved kernel recursive least squares algorithm,” *IEEE transactions on cybernetics*, vol. 49, no. 4, pp. 1160–1172, 2018.

[9] W. Liu, J. C. Principe, and S. Haykin, *Kernel Adaptive Filtering: A Comprehensive Introduction*, vol. 57, John Wiley & Sons, New Jersey, United States, 2011.

[10] W. Weifeng Liu, I. Il Park, and J. C. Principe, “An information theoretic approach of designing sparse kernel adaptive filters,” *IEEE Transactions on Neural Networks*, vol. 20, no. 12, pp. 1950–1961, 2009.

[11] B. Chen, S. Zhao, P. Zhu, and J. C. Príncipe, “Quantized kernel least mean square algorithm,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 1, pp. 22–32, 2011.

[12] J. Q. Candela and O. Winther, “Incremental Gaussian processes,” *Advances in Neural Information Processing Systems*, pp. 1025–1032, 2003.

[13] J. Pardo, F. Zamora-Martínez, and P. Botella-Rocamora, “Online learning algorithm for time series forecasting suitable for low cost wireless sensor networks nodes,” *Sensors*, vol. 15, no. 4, pp. 9277–9304, 2015.

[14] S. Garcia-Vega, X.-J. Zeng, and J. Keane, “Stock returns prediction using kernel adaptive filtering within a stock market interdependence approach,” *Expert Systems with Applications*, vol. 160, Article ID 113668, 2020.

[15] X. Zhong and D. Enke, “Forecasting daily stock market return using dimensionality reduction,” *Expert Systems with Applications*, vol. 67, pp. 126–139, 2017.

[16] S. Jeon, B. Hong, and V. Chang, “Pattern graph tracking-based stock price prediction using big data,” *Future Generation Computer Systems*, vol. 80, pp. 171–187, 2018.

[17] J. Eapen, D. Bein, and A. Verma, “Novel deep learning model with cnn and bi-directional lstm for improved stock market index prediction,” in *Proceedings of the 2019 IEEE 9th annual computing and communication workshop and conference (CCWC)*, pp. 0264–0270, IEEE, Las Vegas, NV, USA, January 2019.

[18] N. Zhang, A. Lin, and P. Shang, “Multidimensionalk-nearest neighbor model based on EEMD for financial time series forecasting,” *Physica A: Statistical Mechanics and Its Applications*, vol. 477, pp. 161–173, 2017.

[19] M. Kumar and M. Thenmozhi, “Forecasting stock index returns using arima-svm, arima-ann, and arima-random forest hybrid models,” *International Journal of Banking, Accounting and Finance*, vol. 5, no. 3, pp. 284–308, 2014.

[20] Y. Qiu, H.-Y. Yang, S. Lu, and W. Chen, “A novel hybrid model based on recurrent neural networks for stock market timing,” *Soft Computing*, vol. 24, pp. 1–18, 2020.

[21] K. George and P. Mutalik, “A multiple model approach to time-series prediction using an online sequential learning algorithm,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 5, pp. 976–990, 2017.

[22] T. Choudhary, V. Mishra, A. Goswami, and J. Sarangapani, “A comprehensive survey on model compression and acceleration,” *Artificial Intelligence Review*, vol. 53, no. 7, pp. 5113–5155, 2020.

[23] E. Hoseinzade and S. Haratizadeh, “CNNpred: CNN-based stock market prediction using a diverse set of variables,” *Expert Systems with Applications*, vol. 129, pp. 273–285, 2019.

[24] M. Kaur and D. Singh, “Multi-modality medical image fusion technique using multi-objective differential evolution based deep neural networks,” *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 2, pp. 2483–2493, 2021.

[25] H. Kaushik, D. Singh, M. Kaur, H. Alshazly, A. Zaguia, and H. Hamam, “Diabetic retinopathy diagnosis from fundus images using stacked generalization of deep models,” *IEEE Access*, vol. 9, pp. 108276–108292, 2021.

[26] C. Sang and M. Di Pierro, “Improving trading technical analysis with tensorflow long short-term memory (lstm) neural network,” *The Journal of Finance and Data Science*, vol. 5, no. 1, pp. 1–11, 2019.

[27] M. Nabipour, P. Nayyeri, H. Jabani, A. Mosavi, E. Salwana, and S. Sahab, “Deep learning for stock market prediction,” *Entropy*, vol. 22, no. 8, p. 840, 2020.

[28] P. Gao, R. Zhang, and X. Yang, “The application of stock index price prediction with neural network,” *Mathematical and Computational Applications*, vol. 25, no. 3, p. 53, 2020.

[29] A. Moghar and M. Hamiche, “Stock market prediction using lstm recurrent neural network,” *Procedia Computer Science*, vol. 170, pp. 1168–1173, 2020.

[30] M. Yukawa, “Multikernel adaptive filtering,” *IEEE Transactions on Signal Processing*, vol. 60, no. 9, pp. 4672–4682, 2012.

[31] J. Fernandez-Bes, V. Elvira, and S. Van Vaerenbergh, “A probabilistic least-mean-squares filter,” in *Proceedings of the 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2199–2203, IEEE, South Brisbane, QLD, Australia, April 2015.

[32] S. Garcia-Vega, X.-J. Zeng, and J. Keane, “Stock price prediction using kernel adaptive filtering within a stock market

interdependence approach," *SSRN Electronic Journal*, vol. 160, p. 113668, 2020.

[33] V. Yadav, A. K. Yadav, M. Kaur, and D. Singh, "Trigonometric mutation and successful-parent-selection based adaptive asynchronous differential evolution," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–18, 2021.

[34] Z. Liu, C. K. Loo, K. Pasupa, and M. Seera, "Meta-cognitive recurrent kernel online sequential extreme learning machine with kernel adaptive filter for concept drift handling," *Engineering Applications of Artificial Intelligence*, vol. 88, Article ID 103327, 2020.

[35] S. Garcia-Vega, X.-J. Zeng, and J. Keane, "Learning from data streams using kernel least-mean-square with multiple kernel-sizes and adaptive step-size," *Neurocomputing*, vol. 339, pp. 105–115, 2019.

[36] W. Liu, P. P. Pokharel, and J. C. Principe, "The kernel least-mean-square algorithm," *IEEE Transactions on Signal Processing*, vol. 56, no. 2, pp. 543–554, 2008.

[37] W. Liu and J. C. Príncipe, "Kernel affine projection algorithms," *EURASIP Journal on Applied Signal Processing*, vol. 2008, pp. 1–12, 2008.

[38] S. Zhao, B. Chen, P. Zhu, and J. C. Príncipe, "Fixed budget quantized kernel least-mean-square algorithm," *Signal Processing*, vol. 93, no. 9, pp. 2759–2770, 2013.

[39] S. Zhao, B. Chen, and J. C. Principe, "Kernel adaptive filtering with maximum correntropy criterion," in *Proceedings of the 2011 International Joint Conference on Neural Networks*, pp. 2012–2017, IEEE, San Jose, CA, USA, June 2011.

[40] S. K. Ahn and G. C. Reinsel, "Estimation for partially nonstationary multivariate autoregressive models," *Journal of the American Statistical Association*, vol. 85, no. 411, pp. 813–823, 1990.

[41] T. Ouyang, H. Huang, Y. He, and Z. Tang, "Chaotic wind power time series prediction via switching data-driven modes," *Renewable Energy*, vol. 145, pp. 270–281, 2020.

[42] A. H. Moghaddam, M. H. Moghaddam, and M. Esfandyari, "Stock market index prediction using artificial neural network," *Journal of Economics, Finance and Administrative Science*, vol. 21, no. 41, pp. 89–93, 2016.

[43] M. R. Hassan, "A combination of hidden Markov model and fuzzy model for stock market forecasting," *Neurocomputing*, vol. 72, no. 16-18, pp. 3439–3446, 2009.

[44] M. Nikou, G. Mansourfar, and J. Bagherzadeh, "Stock price prediction using deep learning algorithm and its comparison with machine learning algorithms," *Intelligent Systems in Accounting, Finance and Management*, vol. 26, no. 4, pp. 164–174, 2019.