Hindawi
Computational Intelligence and Neuroscience
Volume 2021, Article ID 6400045, 14 pages
https://doi.org/10.1155/2021/6400045

Research Article

Multivariate and Online Prediction of Closing Price Using Kernel Adaptive Filtering

Shambhavi Mishra,[1] **Tanveer Ahmed**,[1] **Vipul Mishra**,[1] **Manjit Kaur**,[2] **Thomas Martinetz**,[3] **Amit Kumar Jain**,[4] **and Hammam Alshazly**[5]

[1]*School of Engineering and Applied Sciences, Bennett University, Greater Noida 201310, India*
[2]*School of Electrical Engineering and Computer Science, Gwangju Institute of Science and Technology, Gwangju 61005, Republic of Korea*
[3]*Institute for Neuro- and Bioinformatics, University of Lübeck, Lübeck 23562, Germany*
[4]*Institute for Manufacturing, Cambridge University, Cambridge, UK*
[5]*Faculty of Computers and Information, South Valley University, Qena 83523, Egypt*

Correspondence should be addressed to Hammam Alshazly; hammam.alshazly@sci.svu.edu.eg

Received 2 August 2021; Revised 5 November 2021; Accepted 26 November 2021; Published 17 December 2021

Academic Editor: Radu-Emil Precup

This paper proposes a multivariate and online prediction of stock prices via the paradigm of kernel adaptive filtering (KAF). The prediction of stock prices in traditional classification and regression problems needs independent and batch-oriented nature of training. In this article, we challenge this existing notion of the literature and propose an online kernel adaptive filtering-based approach to predict stock prices. We experiment with ten different KAF algorithms to analyze stocks' performance and show the efficacy of the work presented here. In addition to this, and in contrast to the current literature, we look at granular level data. The experiments are performed with quotes gathered at the window of one minute, five minutes, ten minutes, fifteen minutes, twenty minutes, thirty minutes, one hour, and one day. These time windows represent some of the common windows frequently used by traders. The proposed framework is tested on 50 different stocks making up the Indian stock index: Nifty-50. The experimental results show that online learning and KAF is not only a good option, but practically speaking, they can be deployed in high-frequency trading as well.

1. Introduction

Prediction has applications in a multitude of areas such as economics [1], business planning and production [2], and weather forecasting [3]. However, accurately predicting the value of a variable is one of the very basic and nontrivial problems of the literature. In this article, we focus our attention on financial time-series prediction and its application to stock price forecasting. Stock market is often considered as a chaotic [4], complex [5], volatile [6], and a dynamic mixture of forces driving the movement of a stock. Undoubtedly, its prediction is one of the significant challenges of the literature [7]. Moreover, the Efficient Market Hypothesis [8] states that stock prices reflect all current information, and any new information leads to unpredictability in stock prices. Naturally, significant work has been done in this area. Nevertheless, research clearly specifies that prediction of stocks, especially the nonlinear and nonstationary financial time-series forecasting, is still challenging [9]. In this regard, several models have been developed; for instance, studies focused on volatility [6, 10], option pricing [11], classification of stock movements [12], predicting prices [13], and so on. In addition to this, studies have used a plethora of techniques, for example, support vector machine (SVM) [14], neural network (NN) [15], and genetic algorithm [16]. Nevertheless, a true solution is yet to be found. Moreover, during our literature survey, we found that the paradigm of KAF is not thoroughly investigated. Although, there are a few papers on the topic, e.g., [17, 18], a comprehensive investigation conducted at a large scale eludes the literature. The existing literature focuses on the multiple-kernel learning method and solves different issues such as kernel size and step size. We follow the same line of thought and take the existing methods [17, 19, 20] as the foundation of the proposed work to propose a KAF-based approach for close-price prediction.

We pointed in the previous paragraph that work has ignored KAF as an effective tool for financial time series forecasting. In this context, working with KAF has several advantages. First, it is one most favoured tools of the literature to predict a time series [21, 22]. The techniques in KAF have achieved tremendous accuracy in terms of predictive capability. Second, the convergence speed of KAF-based algorithms is excellent. In other words, they achieve convergence in fewer iterations. Third, they have universal function approximation properties [23]. This has the mathematical property desired for predicting a financial time series. Owing to these reasons, we focus on predicting the financial time series via the paradigm of KAF. Despite these advantages, one of the issues with existing work is Batch Learning. We would argue that Batch Learning is an ineffective tool in financial time-series forecasting. The rationale here is backed by the fact that the data of a financial time series is nonstationary. Therefore, relying on models trained in an offline manner and expecting them to perform well in real market scenarios is a rather strong assumption. To fix this, online learning is proving to be a highly efficient approach [24–27]. In this method, the basis is selected during sample-by-sample training. Moreover, changing circumstances are quickly incorporated, and the algorithm changes its weight vector to make accurate predictions. Hence, we complement the idea of using KAF with online learning to predict a financial time series.

In light of the challenges and the potential solution specified in this section, we propose the paradigm of online KAF for stock price prediction. Thus, this study aims to predict stock movements in an online manner. Although the issue of financial series forecasting is challenging, the goal of this article however is to take one more step towards addressing the issue and to try and lay the groundwork for future work. To do this, we use the National Stock Exchange (NSE), Nifty-50 dataset, which contains 50 leading stocks. In order to describe the contribution of this paper, the following points summarize the essence of the article in brief:

(i) We propose the use of online-KAF techniques for stock price prediction.

(ii) The data is collected at multiple time windows, i.e., one day, sixty minutes, thirty minutes, twenty five minutes, twenty minutes, fifteen minutes, ten minutes, five minutes, and one minute. The proposed idea is applied to each of these time windows to try and find the best window for stock price prediction.

(iii) The main objective is to predict the closing price of a stock. To do that, we apply ten different KAF-based algorithms and present a comprehensive discussion detailing every aspect of the analysis. With numerical testing performed on all fifty stocks of the main index (Nifty-50), we show the work's efficacy in this article.

(iv) We experiment with two different years. First, we try to predict stock prices for the year 2020. Second, we apply the same set of parameters on the most recent data (2021) and try to show the efficacy of the work. Through experiments performed on these two different years, we have found the method proposed in the paper outperforms similar methods in the literature.

(v) Lastly, we also try to show that although the KAF class of algorithms is new in the arena of stock prediction, they nevertheless are a practically viable candidate.

The rest of the paper is organized as follows: In Section 2, we discuss the related work. A discussion on different KAF algorithms is presented in Section 3. The experimental results are described in Section 4. Finally, the conclusion is given in Section 5.

2. Related Work

Stock price prediction is one of the nontrivial problems of the literature. Numerous studies have attempted to explain that stock price prediction is difficult to implement because of inherent nonstationarity in the data [28, 29]. Previous research has shown that stock market prediction is noisy and chaotic and follows nonlinearity [4, 6, 30]. Nonlinear modelling methodologies have been proven to be effective in modelling systems in a variety of domains such as in [31, 32]. Various applications found different modelling approaches to solve nonlinearity problems such as in [33–35]. In traditional prediction, the techniques were based on technical analysis with standards of resistance, support, and indicators using past prices [36]. Previous research has also studied various linear techniques such as moving averages, autoregessive models, discriminating analyses, and correlations [37, 38]. Much of the current literature on stock market prediction pays particular attention to machine learning (ML) techniques. ML has emerged as another popular area for time series prediction. Among the available popular techniques, machine learning methods are researched mostly due to their capabilities for recognizing complex pattern in stock prices [39–42].

Based on the time-varying and nonlinearity aspect of time series, there is a massive demand for online prediction algorithms. It follows the idea of sequential calculation and generates faster and accurate outcomes [26]. To date, various methods have been developed, such as neural network (NN), kernel adaptive filter (KAF) algorithms, and online support vector regression (SVR) [22]. However, neural networks suffer from slow convergence and significant computing needs. In addition, SVR and kernel methods have no problem of falling into the local optimum. SVR has a strong generalization ability [43], but it is just suitable for smaller data. In addition, a multifilter neural network (MFNN) is also used to predict stock price movement. The performance of the MFNN was found to be better than other NN approaches, SVM, and random forests [44]. In [45], the authors combined support vector machines for regression (SVR) and kernel principal component analysis (KPCA) to enhance prediction accuracy that may help investors for short-term decisions. However, the high dimension of input variables makes the learning process long, and the final model computational complexity becomes very large. These machine learning methods had a drawback of large time consumption during the learning process.

To reduce the computational burden, kernel-based online learning algorithms have become gradually popular [44, 46]. In this respect, recurrent kernel online learning is applied to predict the transaction price of specific products. It was observed that the model was stable with a low dependency to parameter settings [47]. Similarly, convolutional neural networks (CNNs) are also suggested for predicting the next-day prices [48]. In all, there is sufficient literature that suggests that modelling the movement of a stock price is nontrivial. In this respect, adaptive filtering has proved to be a standard option for prediction model for streaming data with nonstationary properties [49–51]. KAF can therefore be used for sequential prediction of stock prices by exploiting the market interdependence. KAF are preferred because they are nonparametric, have low computational complexity, and converge very fast [21, 52–55]. In this domain, multiple algorithms are proposed for nonstationary data. They are preferred due to insensitivity towards design parameters [49]. Multistep predictions for stocks using meta-cognitive recurrent kernel online learning is proposed in [56]. The advantage of the KAF method is that it solves various problems in balancing efficiency and prediction accuracy.

Currently, the use of KAF approaches in the stock price prediction is limited [19, 20]. In [19], a multiple-kernel learning method was proposed to address KAF's two main issues: kernel size and step size. In [20], the idea of the local models was proposed to learn the behavior from different stock markets and compare with other online learning methods such as LSTM, quantized kernel least mean square (QKLMS), nearest instance centroid estimation (NICE), vector autoregression (VAR), and vector error-correction model (VECM) for daily closing price prediction. In another research, the study in [57] proposed the idea of adaptive stock trading strategies with deep reinforcement learning methods focused on extracting informative financial features via two methods: gated deep Q-learning trading strategy (GDQN) and gated deterministic policy gradient trading strategy (GDPG). This paper proposes an online KAF-based learning approach. The selection of basis functions can be done during sample-by-sample training in online kernel learning, which is a more efficient option. This method can be incredibly efficient and successful because they only require one pass over the training data.

To the best of our knowledge, the work presented in this article is the first wherein we comprehensively analyze the price of a stock on multiple time windows and comprehensively test the application of the KAF class of algorithms in stocks. Consequently, to discuss the novel contribution, the following points summarize the fundamental differences between this paper and existing work:

(i) To the best of our knowledge, we are the first to use KAF algorithms with multiple time windows to analyze and predict stock prices.

(ii) Stock prediction using existing online methods requires a lot of computation time. The article aims to present a general framework wherein the price prediction can be made in a significantly less amount of time.

(iii) Stock traders can quickly sell and buy specific stocks with numerous time windows using the proposed strategy, resulting in larger earnings.

3. Methodology

As discussed in Section 1, we have worked with KAF-based techniques. Furthermore, we use online prediction methods. In this regard, KAFs work by self-tuning, where the input-output mapping is formulated according to an optimization criterion usually determined by the error signal. There are two types of adaptive filters: linear and nonlinear. In linear filters, the traditional system follows a supervised technique and depends upon error-correction learning. The filter adaptively adjusts weights, $\omega(i-1)$, where i denotes the discrete-time interval. Here, the input signal v_i is mapped to an actual response t_i. Correspondingly, an error is denoted by e_i. The error signal adjusts weights by incremental value denoted by $\Delta\omega_i$. At the next iteration, ω_i becomes the current value of the weight to be updated. This process is continuously repeated until the filter reaches convergence; this generally occurs when the weight adjustment is small enough. Linear adaptive filters do not give satisfactory performance for the nonlinear system due to the results varied in a nonintuitive manner. In real-world problems, where data patterns are more complex, classes may not be separated easily by hyperplanes. Consequently, we have to look to nonlinear methods. In this paradigm, data is projected into high-dimensional linear feature space and prediction is done in this high-dimensional space. Comparing with other existing techniques for regression and classification, KAF has the following advantages:

(i) KAFs are universal approximators.

(ii) KAFs handle the complexity issues in terms of computation and memory. Moreover, they follow the property of no local minima.

(iii) KAFs follow the idea of online learning and handle nonstationary conditions well.

It was discussed that nonlinear adaptive techniques are well suited for real-world problems. In this regard, kernel methods transform data into a set of points in the RKHS (Reproducing Kernel Hilbert Space). The main idea of KAF can be summarized as the transformation of input data into a high-dimensional feature space G, via Mercer kernel. For this, the problem can be solved via inner products. There is no need to do expensive computations in high-dimensional space, owing to the famous "kernel trick." Considering KAF, suppose we have an input-output mapping as $g\colon \mathscr{V} \longrightarrow R$, based on a well-known sequence $((v_1, t_1), (v_2, t_2), \ldots, (v_i, t_i))$. Here, v_i is the system input with $i = 1, \ldots, n$ and t_i is equivalent to desired response. The goal is to estimate g from data. In KAFs, generally, the computation involves the use of a kernel. An example of a kernel is given as follows:

$$\kappa\langle v, v'\rangle = \exp\frac{\left(\|v - v'\|^2\right)}{\sigma^2}. \tag{1}$$

Here, κ denotes the kernel and σ denotes the kernel width.

3.1. Discussion on KAF Algorithms. In this section, we discuss some of the most popular methods in KAF. For reasons of brevity, we keep the discussion short.

3.1.1. Least Mean Square (LMS) Algorithm. According to [46], the main aim of LMS algorithm is to minimize the following empirical risk function:

$$\min_{\omega} R_{\text{emp}}\left[\omega \in H_1, R^L\right] = \sum_{i=1}^{N} \left(t_i - \omega(v_i)\right)^2. \tag{2}$$

Applying stochastic gradient descent (SGD), equation (2) can be represented as

$$\begin{aligned} \omega_0 &= 0, \\ e_i &= t_i - \omega_{(i-1)} v_i, \\ \omega_i &= \omega_{(i-1)} + \eta e_i v_i, \end{aligned} \tag{3}$$

where η is step size and e_i is known as prior error.

The weight-update equation results in the following form:

$$\omega_i = \eta \sum_{i=1}^{N} e_i v_i. \tag{4}$$

Representing the idea in terms of inner product, we get

$$\begin{aligned} t = \omega_i(v) &= \eta \sum_{i=1}^{n} e_i \langle v_i, v\rangle, \\ e_i &= t_i - \eta \sum_{i=1}^{n-1} e_i \langle v_i, v\rangle. \end{aligned} \tag{5}$$

3.1.2. Kernel Least Mean Square Algorithm (KLMS). KLMS [21] is an extension of LMS algorithm, the main difference is input v_i is transformed to $\Psi(v_i)$ in the high-dimensional space RKHS. Applying LMS algorithm at new sequences $\{\Psi(i), t_i\}$, we get

$$\begin{aligned} \omega_0 &= 0, \\ e_i &= t_i - \omega_{(i-1)}^T \Psi(i), \\ \omega_i &= \omega_{(i-1)} + \eta e_i \Psi(i), \end{aligned} \tag{6}$$

where e_i is the prediction error, ω_i is the weight vector in G, and η is the step size.

Using the kernel trick, KLMS can now be written as

$$\begin{aligned} g_0 &= 0, \\ e_i &= t_i - g_{i-1}(v_i), \\ g_i &= g_{i-1} + \eta e_i \kappa\langle v_i, .\rangle. \end{aligned} \tag{7}$$

KLMS assigns new unit for every input v_i as the center with ηe_i as its coefficient. Following the radial basis function (RBF), the algorithms are represented as follows:

$$g_i = \sum_{j=1}^{i} b_j(i) \kappa\langle v_j, .\rangle. \tag{8}$$

3.1.3. Kernel Recursive Least Square Algorithm (KRLS). According to [21], in KRLS, the objective function is complemented via a regularization parameter. This can be represented as follows:

$$\min_{\omega} \sum_{j=1}^{i} \left|t(j) - \omega^T \Psi(j)\right|^2 + \Lambda\|\omega\|^2, \tag{9}$$

where Λ stands for regularization vector.

It is shown that $\omega_i = \psi(i) b(i)$, where $b(i) = [\Lambda I + L(i)]^{-1} t_i$; also, $t_i = [t_1, t_2, t_3 \ldots, t_i]^T$, $L(i) = \psi(i)^T \psi(i)$, and $\psi(i) = [\Psi(1), \Psi(2), \Psi(3), \ldots, \Psi(i)]$.

Complementing the previous equation with RBF, we get

$$g_i = \sum_{j=1}^{i} b_j(i) \kappa\langle v_j, .\rangle. \tag{10}$$

The whole idea here can now be summarized as

$$\begin{aligned} R(i-1) &= (\Lambda I + L(i-1))^{-1}, \\ O(i) &= \psi(i-1)^T \Psi(i), \\ E(i) &= R(i-1) O(i), \\ U(i) &= \Lambda + \kappa\langle v_i, v_i\rangle - E(i)^T O(i). \end{aligned} \tag{11}$$

Following the sequential property of KRLS, we have

$$\begin{aligned} g_0 &= 0, \\ e_i &= t_i - g_{i-1}(v_i), \\ g_i &= g_{i-1} + U(i)^{-1} e_i \kappa\langle v_i, .\rangle - \sum_{j=1}^{i-1} U(i)^{-1} e_i E_j(i) \kappa\langle v_j, .\rangle. \end{aligned} \tag{12}$$

KRLS updates all previous coefficients through $-U(i)^{-1} e_i E(i)$, whereas KLMS never updates previous coefficients. Here, $E_j(i)$ is the j^{th} component of $E(i)$. The computational complexity of KRLS is $O(i^2)$.

3.1.4. Kernel Affine Projection Algorithms (KAPAs). KAPA [58] derives the idea of KLMS while reducing boosting performance and gradient noise. In KAPA, we formulate with sequences $\{t_1, t_2\}$ and $\{\Psi(1), \Psi(2)\}$ to minimize the cost function and estimate with weight vector ω.

$$\min_{\omega} \text{emp}\left|t - \omega^T \Psi(v)\right|^2. \tag{13}$$

Using stochastic gradient descent, we replace covariance matrix and cross covariance vector by local approximation directly from the data. Hence, we get the following equations:

$$\omega_i = \omega_{(i-1)} + \eta\psi(i)\left[t(i) - \psi(i)^T \omega_{(i-1)}\right], \tag{14}$$

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

3.1.5. Quantized Kernel Least Mean Square Algorithm (QKLMS). QKLMS is a famous algorithm proposed in [50]. It is an extension of KLMS algorithm to deal with the issue of data redundancy. Using quantization operator the core idea can be written as

$$\begin{aligned} \omega_0 &= 0, \\ e_i &= t_i - \omega_{(i-1)}^T \Psi(i), \\ \omega_i &= \omega_{(i-1)} + \eta e_i \mathcal{Q}[\Psi(i)], \end{aligned} \tag{15}$$

where, in feature space G, $\mathcal{Q}[.]$ denotes the quantization. The learning rule for QKLMS is

$$\begin{aligned} g_0 &= 0, \\ e_i &= t_i - g_{i-1}(v_i), \\ g_i &= g_{i-1} + \eta e_i \kappa(Q[v_i], .). \end{aligned} \tag{16}$$

QKLMS and KLMS have almost the same computational complexity. The only difference between the two algorithms is that QKLMS deal with the issue of data redundancy to locally update the coefficients of closest center.

In short, the central theme of QKLMS is given in Algorithm 1.

3.1.6. Kernel Normalized Least Mean Square Algorithm (KNLMS). According to [49], KNLMS algorithm is used for dictionary designing with coherence criterion. Here, we discuss KNLMS from the point of view of MKNLMS-CS (multikernel normalized least mean square algorithm with coherence based sparsification).

Assume κ_m: $\mathcal{V} \times \mathcal{V} \longrightarrow R$, where $m \in \mathcal{M}$: $\{1, 2, \ldots, M\}$ is a set of M distinct kernels.

Consider $\mathcal{J}_n^{cs} := j_1^{(n)}, j_2^{(i-1)}, \ldots, j_{r_n}^{(i-1)} \subset \{0, 1, \ldots, n-1\}$ to be the dictionary $\left\{\kappa_m(., v_j)\right\}_{m\in\mathcal{M}_{j\in\mathcal{J}_n^{cs}}}$.

Here, $r_n := |\mathcal{J}_n^{cs}|$ is the size of dictionary. The filter works as per the following set of rules:

$$\Psi_n^{cs}(v) = \sum_{m\in\mathcal{M}} \sum_{j\in\mathcal{J}_n^{cs}} h_{j,n}^{(m)} \kappa_m\left(v, v_j\right), \quad v \in \mathcal{V}, \tag{17}$$

where $h_{j,n}^{(m)} \in R, m \in \mathcal{M}, j \in \mathcal{J}_n^{cs}$. The estimated error $\hat{t}_n := \Psi_n^{cs}(v_n)$ of t_n can be written as

$$\Psi_n^{cs}\left(v_n\right) = \sum_{j\in\mathcal{J}_n^{cs}} h_{j,n}^T \kappa_{j,n}, \tag{18}$$

where

$$\begin{aligned} \kappa_{j,n} &:= \left[\kappa_1\left(v_n, v_j\right), \kappa_2\left(v_n, v_j\right), \ldots, \kappa_M\left(v_n, v_j\right)\right]^T \in R^M, \\ h_{j,n} &:= \left[h_{j,n}^{(1)}, h_{j,n}^{(2)}, h_{j,n}^{(3)}, \ldots, h_{j,n}^{(M)}\right]^T \in R^M. \end{aligned} \tag{19}$$

Let the initial dictionary be indicated as $\mathcal{J}_0^{cs} := \varnothing$. This makes H_0 to be an empty size matrix. Following algorithm, we only add a new point n into $\mathcal{J}_n^{cs}$ if the following condition holds:

$$\|\kappa\|_{\max} := \max_{m\in\mathcal{M}} \max_{j\in\mathcal{J}_n^{cs}} \left|\kappa_m\left(v_n, v_j\right)\right| \leq \Delta, \quad n \in N, \tag{20}$$

where $\Delta > 0$ is the threshold. Let $\eta \in [0, 2]$ denote the step size and $\Lambda > 0$ denote the regularization parameter. The update rule is given as follows:

(i) If equation (20) is satisfied, $\mathcal{J}_{n+1}^{cs} := \mathcal{J}_n^{cs} \cup \{n\}$. Also,

$$H_{n+1} := \overline{H}_n + \eta \frac{t_n - \langle \overline{K}_n, \overline{H}_n \rangle}{\left\|\overline{K}_n\right\|^2 + \Lambda} \overline{K}_n. \tag{21}$$

(ii) If equation (20) is not satisfied, $j_{n+1}^{cs} := j_n^{cs}$. Also,

$$H_{n+1} := H_n + \eta \frac{t_n - \langle H_n, K_n \rangle}{\left\|K_n\right\|^2 + \Lambda} K_n, \tag{22}$$

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

3.1.7. Probabilistic Least Mean Square Algorithm (PROB-LMS). The probabilistic approach to the LMS filter is an efficient approximation method. It provides an adaptable step-size LMS algorithm together with a measure of uncertainty about the estimation. In addition, it also preserves the linear complexity of the standard LMS. Some of the advantages of probabilistic models are as follows: (1) they force the designer to specify all the assumptions of the model, (2) they provide a clear separation between the model and the algorithm used to solve it, and (3) they usually provide some measure of uncertainty about the estimation. It is assumed observation models to be Gaussian with this distribution:

$$pr\left(t_k \mid \omega_k\right) = \mathcal{N}\left(t_k; v_k^T \omega_k, \sigma_n^2\right), \tag{23}$$

```
    **Initialization:** Determine quantization size $\varepsilon_{\mathcal{V}} \geq 0$, step size $\eta > 0$, kernel parameter $\sigma > 0$.
    **Output:** The center set and coefficient vector: $D(1) = \{v_1\}$, $b_1 = [\eta t_1]$
    **Conditions**:
    **while** $\{v_i, t_i\}\,(i > 1)$ is available
(i)   Calculate the result of the adaptive filters as
      $y_i = \sum_{j=1}^{\mathrm{size}(D(i-1))} b_j(i-1)\kappa(D_j(i-1), v_i)$
(ii)  Calculate the error between actual and desired output $e_i = t_i - y_i$
(iii) Calculate the distance between $v_i$ and $D(i-1)$
      $\mathrm{distance}(v_i, D(i-1)) = \min_{1 \leq j \leq \mathrm{size}(D(i-1))} \|v_i - D_j(i-1)\|$
(iv)  If the distance $(v_i, D(i-1)) \leq \varepsilon_{\mathcal{V}}$ then codebook does not require any changes: $D(i) = D(i-1)$. Quantize $v_i$ to the closest code
      vector: $b_{j^*}(i) = b_{j^*}(i-1) + \eta e_i$
      where, $j^* = \mathrm{argmin}_{1 \leq j \leq \mathrm{size}(D(i-1))} \|v_i - D_j(i-1)\|$,
(v)   Otherwise update the codebook with new center and update center $D(i) = \{D(i-1), v_i\}$ and $b_i = [b(i-1), \eta e_i]$.
    **end while**
```

Algorithm 1: The central idea of the QKLMS algorithm.

where ω_k = parameter vector, σ_n^2 = variance for observation noise, and v_k = regression vector.

3.1.8. Kernel Maximum Crossentropy Criterion (KMCC). The algorithm's main aim is to maximize crossentropy between desired t_i and actual output y_i [55]. Using MCC criterion and SGD, the algorithm can be written as

$$\begin{aligned}
\omega_0 &= 0, \\
\omega_{(i+1)} &= \omega_i + \eta \frac{\partial \kappa_\sigma\left(t_i, \omega_i^T \Psi(v_i)\right)}{\partial \omega_i} \\
&= \omega_i + \eta \left[\left(\exp \frac{\left(-e_i^2\right)}{2\sigma^2}\right) e_i \Psi(i)\right] \ldots \\
&= \eta \sum_{i=1}^{n} \left[\left(\exp \frac{\left(-e_i^2\right)}{2\sigma^2}\right) e_i \Psi(i)\right],
\end{aligned} \tag{24}$$

where σ is the kernel width and η is the step size.

The complete prediction and error calculation can be summarized as

$$\begin{aligned}
y_i &= \eta \sum_{i=1}^{n} \left[\left(\exp \frac{\left(-e^2\right)}{2\sigma^2}\right) e_i \kappa\langle v_i, v_n\rangle\right], \\
e_i &= t_i - y_i.
\end{aligned} \tag{25}$$

3.1.9. Leaky Kernel Affine Projection Algorithm (LKAPA). The LKAPA [58] is the extension of KAPA discussed in Section 3.1.4. According to equation (14), weight updation is a difficult task in high-dimensional space. Here equation (14) is modified. This can be done as follows:

$$\begin{aligned}
\omega_0 &= 0, \\
\psi(i)^T \omega_{(i-1)} &= \left[\sum_{j=1}^{i-1} b_j(i-1)\kappa_{i-K+1,j}, \ldots, \sum_{j=1}^{i-1} b_j(i-1)\kappa_{i-1,j}, \sum_{j=1}^{i-1} b_j(i-1)\kappa_{i,j}\right]^T, \\
e_i &= t_i - \psi(i)^T \omega_{(i-1)}, \\
\omega_i &= \omega_{(i-1)} + \eta \psi(i) e_i \\
&= \sum_{j=1}^{i=1} b_j(i-1)\Psi(j) + \sum_{j=1}^{K} \eta e_j(i)\Psi(i-j+K),
\end{aligned} \tag{26}$$

where $\kappa_{i,j} = \kappa(v(i), v(j))$

The weight vector is computed using the following criterion:

$$\omega_i = \sum_{j=1}^{i} b_j(i)\Psi(i), \quad \forall_i \geq 0. \tag{27}$$

From the perspective of empirical risk minimization, we minimize the following objective function:

$$\min_{\omega} \text{emp}\left|t - \omega^T \Psi(v)\right|^2 + \Lambda\|\omega\|^2. \quad (28)$$

Then, we get

$$\omega_i = (1 - \Lambda\eta)\omega_{(i-1)} + \eta\psi(i)\left[t(i) - \psi(i)^T \omega_{(i-1)}\right], \quad (29)$$

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

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

$$b_\kappa(i) = \begin{cases} \eta\left(t_i - \sum_{j=1}^{i-1} b_j(i-1)k_{i,j}, \quad k = i,\right. \\ \text{for}\{i - K + 1 \le k \le i - 1\}, \\ (1 - \Lambda\eta)b_k(i-1) + \eta\left(t(k) - \sum_{j=1}^{i-1} b_j(i-1)\kappa_{k,j}\right), \\ 1 \le k < i - K + 1, (1 - \Lambda\eta)b_k(i-1). \end{cases} \quad (30)$$

3.1.10. Normalized Online Regularized Risk Minimization Algorithm (NORMA). NORMA [58] is one of the kernel-based version of LKAPA described in Section 3.1.9. It is also correlated with the KLMS algorithm summarized in Section 3.1.2. NORMA includes the regularization and nonlinear functional approach. It allows to reject old values ones in a sliding window manner.

3.2. Problem Formulation. In this section, we discuss the results of stock prediction using all the ten discussed algorithms. The purpose of stock prediction is to determine the future values of a stock depending upon the historical values. As discussed in the Introduction section, our main aim is to predict the close price. To this end, we calculated the percentage change in close price. Subsequently, we apply the idea of autoregression of the order m to predict the future change in the stock price. An autoregressive (AR) model forecasts future behavior using data from the past. When there is a correlation between the values in a time series and the values that precede and succeed them. In such situations, AR models have shown tremendous potential. In context of the work presented here, the problem is formulated as

$$\Psi(V_i) = \sum_{i=1}^{m} \omega_{(i-1)}\Psi(V_i). \quad (31)$$

$\Psi(V_i)$ is the close price in the high-dimensional space, and ω is the weight vector. Since we follow the AR model, it is imperative to estimate the weight vector. To estimate the weight vector, KAF techniques discussed in the previous section are used. A sample of the formulation is shown in Table 1. In this table, we have shown problem formulation by considering the daywise closing price. This type of procedure is followed commonly in multivariate time series prediction, e.g., [59, 60]. It should be noted here that the procedure was followed for all the window sizes. Subsequently, the problem became autoregression-based next percentage prediction. The actual closing can then be computed from the percentage change easily. The overall framework followed in the article is shown in Figure 1. The experiments were performed on the Nifty-50 dataset, and the data used in the experimentation is available at shorturl.at/lnvF2. The kernel adaptive filtering (KAF) algorithms that were used in this work is available at https://github.com/steven2358/kafbox.

4. Experimental Results

4.1. Dataset Description. In this section, we have described the experimental details of the Nifty-50 dataset. Nifty-50 is the largest stock exchange in India according to the rate of total and average daily turnover for equity shares. We collected the data of all stocks from 9 : 15 to 3 : 30. In addition, we collected the data for two different periods of years. First, we try to predict stock prices for the year 2020 from January 01, 2020, to December 31, 2020, and second, from the most recent data (2021) between January 01, 2021, and May 31, 2021. The original data was available for one-minute open, high, low, and close (OHLC) prices. From this granular data, we clubbed the OHLC quotes to get the data for other time windows. In particular, we created and preprocessed the dataset according to nine prediction windows (one day, sixty minutes, thirty minutes, twenty five minutes, twenty minutes, fifteen minutes, ten minutes, five minutes, and one minute). Recall that we focused on predicting the percentage changes in close price. To that end, we also normalized the data between the range of 0 to 1. Then, ten distinct KAF algorithms were applied to the final preprocessed data for every stock. Finally, it is worth noting that the experimental findings obtained with the KAF algorithm on the Nifty-50 dataset demonstrate the work's superiority and could serve as a new benchmark for the field's future state of the art.

4.2. Evaluation Criterion. To evaluate and compare the performance of various KAF algorithms, we use standard error evaluation metrics such as mean squared error (MSE), mean absolute error (MAE), and directional symmetry (DS). The metrics are elaborated in the following text.

4.2.1. Minimum Square Error (MSE). MSE is also known as mean squared deviation (MSD) which calculates the average squared difference between the actual and predicted observation:

$$\mathbf{MSE} = \sum_{i=1}^{n} (a_i - p_i)^2. \quad (32)$$

4.2.2. Mean Absolute Error (MAE). MAE calculates the average magnitude between actual and predicted observations in a set of predictions, without observing their directions, i.e., the average prediction error.

Table 1: Time window of 1 day (stock-Reliance).

Day	Actual price	Change in price
1 day	1987.5	0.1685
2 day	1990.85	−1.2431
3 day	1966.1	−2.6372
4 day	1914.25	−0.16194
5 day	1911.15	1.17991
6 day	1933.7	−1.8849
7 day	1897.25	3.1519
8 day	1957.05	−0.9325
9 day	1938.8	1.1244
10 day	1960.6	—

If we choose $M = 3$, then **Input** = [{0.1685,-1.2431,-2.6372}] and **Output** = [{−0.16194}].

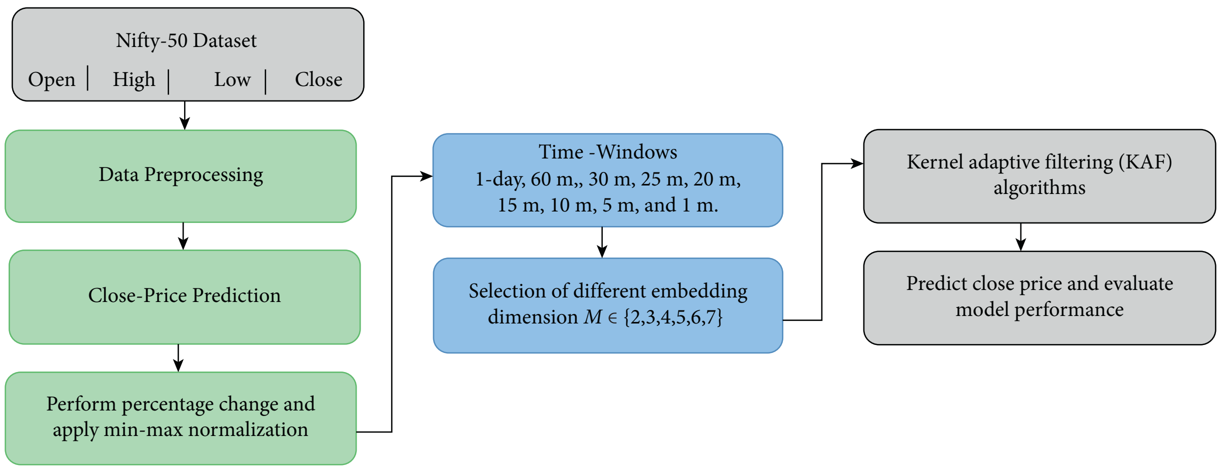

Nifty-50 Dataset
Open
High
Low
Close
Data Preprocessing
Close-Price Prediction
Perform percentage change and apply min-max normalization
Time -Windows 1-day, 60 m,, 30 m, 25 m, 20 m, 15 m, 10 m, 5 m, and 1 m.
Selection of different embedding dimension $M \in \{2,3,4,5,6,7\}$
Kernel adaptive filtering (KAF) algorithms
Predict close price and evaluate model performance

Figure 1: Proposed close price prediction framework.

$$\mathbf{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|a_i - p_i\right|. \tag{33}$$

4.2.3. Directional Symmetry (DS). Directional symmetry in terms of time series analysis measures the model's performance to predict positive and negative trends from one time period to the next.

$$\mathbf{DS} = \frac{1}{n}\sum_{i=1}^{n} d_i, \tag{34}$$

where

$$d_i = \begin{cases} 0, & \text{otherwise}, \\ 1, & \left(a_i - a_{i-1}\right)\left(p_i - p_{i-1}\right) \geq 0, \end{cases} \tag{35}$$

where n is the time-step, a_i represents the actual values, and p_i represents the predicted output. In the following procedure, we discuss the details to compute error values.

4.3. Procedure: Error Computation.

(1) We worked with Nifty-50 firms with 2020 and 2021 datasets, as mentioned in Section 1. Moreover, it was also pointed out that we work with ten different algorithms. The parameters listed in Table 2 were tuned manually. In order to find the optimal values of the parameters, multiple experiments were performed.

(2) To compute the error values for each stock and every algorithm, we formulated the problem as an autoregressive problem (see Section 3.2) and computed the error values for all 50 stocks. In total, we get 50X3 error values, one for MSE, MAE, and DS. Moreover, we pointed out that we have nine different prediction windows. Hence, error estimation was done for all stocks, all windows, and all ten algorithms.

(3) Subsequently, for a particular algorithm, and for a single time window, we take the average of all 50-error metrics (one for every stock) to come up with the final number. The number is presented in the article. This number shows the overall predictive capability of the model on all fifty stocks.

4.4. Prediction, Convergence, and Residual Analysis. In this section, we analyze the performance of KAF algorithms for close price prediction. In this regard, the prediction graph for one stock (Reliance) with KRLS is considered. Figure 2

Table 2: Parameter description for close price using ten different KAF algorithms.

Parameter	KAPA	KLMS	KMCC	KNLMS	KRLS	LKAPA	LMS	NORMA	PROB-LMS	QKLMS
(σ)	4.0	4.0	4.0	4.0	3.0	5.0		7.0		3.0
(η)	1.7	1.1	1.5	1.7		0.09		1.1		1.2
(ε)	1E-4			1E-2						0.3
(Λ)						1E-4		1E-2	0.4	
$(\sigma_2 n)$									2	
$(\sigma_2 d)$									6	
mu0	0.2			2			0.2			
(P)	20					20				
nu					1E-2					
τ								500		
tcoff								0.9		

σ = kernel width, $\sigma_2 n$ = variance of observation noise, $\sigma_2 d$ = variance of filter weight diffusion, η = step size, ε = regularization parameter, Λ = Tikhonov regularization, tcoff = learning rate coefficient, τ = memory size (terms retained in truncation), mu0 = coherence criterion threshold, P = memory length, and nu = approximate linear dependency (ALD) threshold.

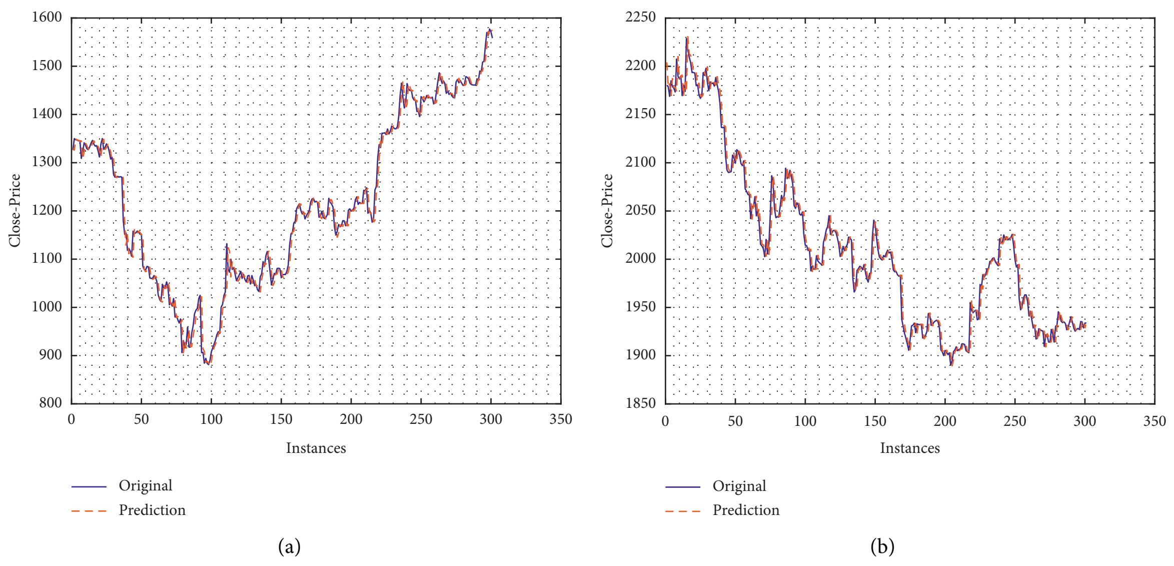
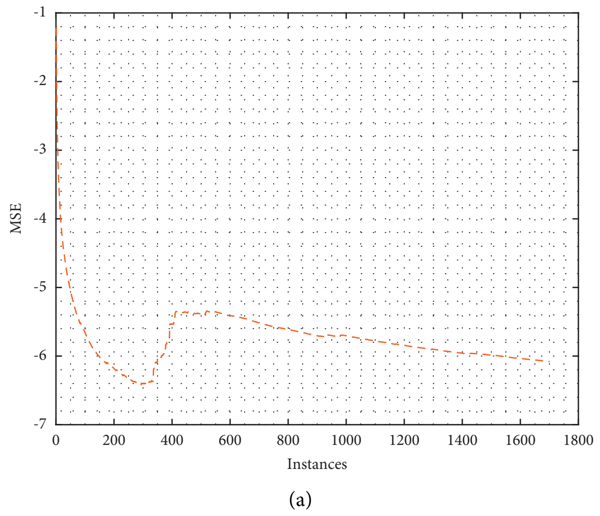


Figure 2: Prediction for one stock (Reliance) using KRLS: (a) 2020 dataset and (b) 2021 dataset.

shows the results for 2020 and 2021 datasets. It is visible from the figure that we are getting good results. It should be noted here that we have presented the result for one stock (Reliance) and one prediction window (sixty minutes). Similar results were obtained for other companies in the dataset. It is also visible from the figure that although the prediction is not cent percent accurate, it is close. It, therefore, implies the superior performance of KAF algorithms in prediction. It should be noted here that although we are getting good results, there are always chances of overfitting. In this article, since we are using online learning, the architecture itself naturally minimizes the chances of overfitting, but it is possible that the superior results might be due to overfitting.

It is expected from any machine learning algorithm that it should converge as we train the model with more instances; in other words, the error as we progress through the training should decrease to an acceptable range. In this regard and in addition to presenting the results for prediction in Figure 2, we have also presented the result of convergence in Figure 3 for 2020 and 2021 datasets, respectively. Similar to the previous case, we have only plotted the result considering single stock (Reliance) and one prediction time window. The convergence graphs of the algorithm were plotted taking MSE as the error metrics. Figure 3 shows the error convergence graph for both the datasets and KRLS algorithm for the Reliance stock. In Figure 3, x-axis shows the number of instances and y-axis shows the MSE. It can be seen from Figure 3 that the algorithm reached convergence very quickly. In fact, the algorithm reached convergence at 1000th data point. Convergence is very important in KAF as it shows the ability of the algorithm to adapt itself and learn from the data quickly. Though, there are minor fluctuations in the end, but it nevertheless is acceptable as there will always be minor changes in the new data.

To complement the prediction results, we have also presented the distribution of error residuals in Figure 4 for 2020 and 2021 datasets, respectively. As visible from the



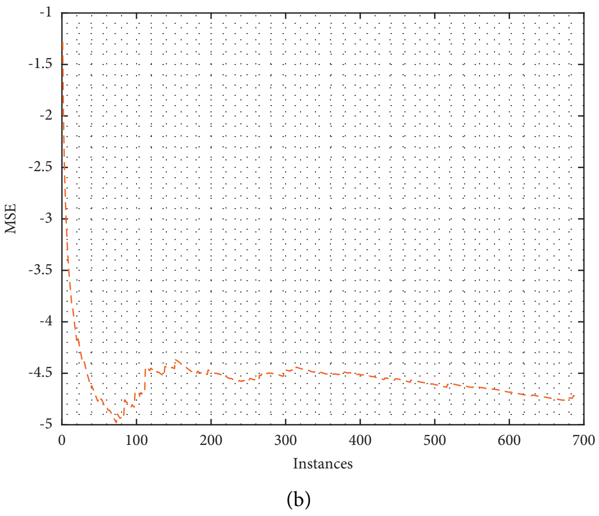
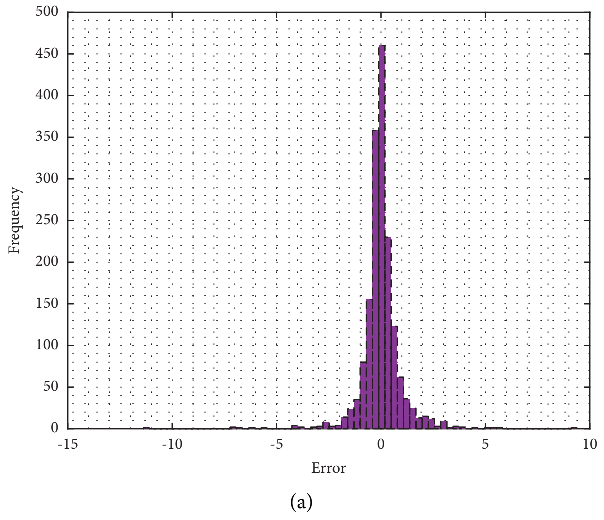


Figure 3: Error convergence for one stock (Reliance) using KRLS: (a) 2020 dataset and (b) 2021 dataset.



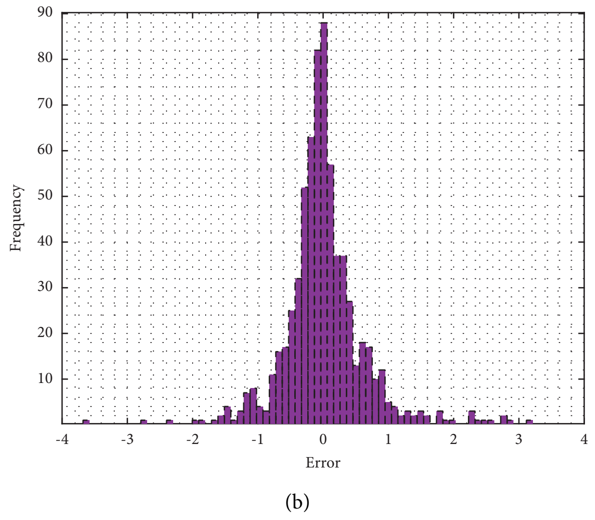


Figure 4: Error residuals for one stock (Reliance) using KRLS: (a) 2020 dataset and (b) 2021 dataset.

figure, the residuals are following a normal distribution. This type of behavior is excellent as there are very few outliers. Moreover, the overall variance of the residuals is also less, showing the excellent prediction potential of the algorithm.

4.5. Comprehensive Evaluation of KAF Algorithms. In contrast to batch learning techniques, which generate the best predictor by learning on the full training dataset at once, we employ an online learning concept in which data becomes available in a sequential order (sample by sample training) and is used to update the best predictor for future data at each step. As we have used ten different algorithms, it is logical to compare the performance of all algorithms. In this regard, we have shown the result in two different datasets. First, we attempt to forecast stock prices for 2020. Second, we use the same set of parameters on the most recent data (2021) to demonstrate the work's efficacy. To evaluate the performance of KAF-based methods, we try to experiment with different values of M (the embedding dimension). We vary the underlying dimensions from 2 to 7 with a step size of 1, i.e., $M \in \{2, 3, 4, 5, 6, 7\}$. With this setup, the results are presented in Tables 3 and 4. It is visible from the table that once again, KRLS performed well in terms of error minimization. The best number for the embedding dimension is two when we consider MSE and MAE. However, when it came to DS, the numbers and the algorithms are different because a market trend is a term used to describe how a market moves over time. A trend can generally move upward or downward. For instance, considering daily data (1 day in the table), the best performing algorithm is LKAPA with embedding dimension $(M) = 5$. In fact, for this metric (DS), we see much conflict in terms of the best algorithm. Nevertheless, the experimentation revealed the superiority of KRLS, PROB-LMS, and LKAPA.

Table 3: Best embedding dimension for all time windows according to evaluation metrics MSE, MAE, and DS (2020).

Time window	MSE	Best embedding dimensions	Algorithms
1 day	0.014 3	2	KRLS
60 minutes	0.003 4	2	KRLS
30 minutes	0.002 0	2	KRLS
25 minutes	0.001 8	2	KRLS
20 minutes	0.001 5	2	KRLS
15 minutes	0.001 2	2	KRLS
10 minutes	0.000 8	2	KRLS
5 minutes	0.000 4	2	KRLS
1 minute	0.000 10	2	KRLS
Time window	MAE	Best embedding dimensions	Algorithms
1 day	2.132 4	2	KRLS
60 minutes	0.660 2	2	KRLS
30 minutes	0.466 6	2	KRLS
25 minutes	0.431 7	2	KRLS
20 minutes	0.381 2	2	KRLS
15 minutes	0.329 5	2	KRLS
10 minutes	0.266 7	2	KRLS
5 minutes	0.188 6	2	KRLS
1 minute	0.085 2	2	KRLS
Time window	DS	Best embedding dimensions	Algorithms
1 day	0.501 3	5	LKAPA
60 minutes	0.491 9	5	KRLS
30 minutes	0.491 3	5	KRLS
25 minutes	0.492 5	3	KRLS
20 minutes	0.488 1	2	PROB-LMS
15 minutes	0.489 3	7	QKLMS
10 minutes	0.490 2	2	KRLS
5 minutes	0.491 0	2	PROB-LMS
1 minute	0.471 5	2	KRLS

Table 4: Best embedding dimension for all time windows according to evaluation metrics MSE, MAE, and DS (2021).

Time window	MSE	Best embedding dimensions	Algorithms
1 day	0.034 2	2	KRLS
60 minutes	0.008 1	2	KRLS
30 minutes	0.004 7	2	KRLS
25 minutes	0.004 2	2	KRLS
20 minutes	0.003 3	2	KRLS
15 minutes	0.002 8	2	KRLS
10 minutes	0.002 0	2	KRLS
5 minutes	0.001 1	2	KRLS
1 minute	0.000 3	2	KRLS
Time window	MAE	Best embedding dimensions	Algorithms
1 day	1.690 1	2	KRLS
60 minutes	0.548 0	2	KRLS
30 minutes	0.396 1	2	KRLS
25 minutes	0.367 1	2	KRLS
20 minutes	0.323 8	2	KRLS
15 minutes	0.280 3	2	KRLS
10 minutes	0.225 9	2	KRLS
5 minutes	0.160 1	2	KRLS
1 minute	0.072 9	2	KRLS
Time window	DS	Best embedding dimensions	Algorithms
1 day	0.487 0	4	NORMA
60 minutes	0.493 0	4	KNLMS
30 minutes	0.484 9	4	KRLS
25 minutes	0.487 8	6	LKAPA
20 minutes	0.489 1	7	LKAPA
15 minutes	0.488 1	2	PROB-LMS
10 minutes	0.489 1	2	PROB-LMS
5 minutes	0.484 6	2	PROB-LMS
1 minute	0.475 1	2	KRLS

4.6. Comparison with Other State-of-the-Art Methods. We compared our result with other learning methods such as [61–63], among other learning approaches. The deep learning (DL) algorithms were taught and assessed over a period of 25 epochs utilizing an 80 : 20 split. The amount of time taken to train and make prediction was recorded. Based on the architecture details and hyperparameter settings provided in the relevant articles, the DL-based method [61–63] stocks were reimplemented. All of the techniques were trained on the Nifty-50 dataset. We chose fifty equities for the sixty-minute time periods to maintain uniformity across different ways for experimentation. In terms of MSE, RMSE, and execution time, all of the approaches were then compared to the suggested KAF method (KRLS). For the 2020 and 2021 datasets, Tables 5 and 6 summarize the comparative outcomes learning approaches. The results in Tables 5 and 6 show that the proposed approach outperforms previous stock prediction methods in the literature.

We must point it out here that since all the models belong to the same category of kernel adaptive filtering, the complexity of all the models is almost similar. For neural networks used in the article, we collect the architecture from their respective papers [61–63]. It should be noted here that KAF is also analogous to the neural network architecture with a single layer. Furthermore, even though it has a single layer, it is giving good results.

Table 5: Comparison of the proposed work with other state-of-the-art stock prediction method for 60-minute time window (2020 dataset) from January 01, 2020, to December 31, 2020.

Method	MSE	RMSE	Execution time (s)
Gao et al. [61]	0.519 17	0.7205	400.39
Moghar et al. [62]	0.518 00	0.7197	1265.11
Nikou et al. [63]	0.518 38	0.7199	5006.19
Proposed method	0.003 4	0.0583	5.234

Table 6: Comparison of the proposed work with other state-of-the-art stock prediction method for 60-minute time window (2021 dataset) from January 01, 2021 to May 31, 2021.

Method	MSE	RMSE	Execution time (s)
Gao et al. [61]	0.702 02	0.8378	362.67
Moghar et al. [62]	0.697 5	0.8351	1082.90
Nikou et al. [63]	0.702 32	0.8380	2250.87
Proposed method	0.008 1	0.09	4.256

Table 7: Effect of dictionary size.

Dictionary size	MSE	MAE	DS	Execution time (seconds)
500 dictionary size	0.020 2	0.687	0.511	0.675
1000 dictionary size	0.019 3	0.674	0.497	0.696
5000 dictionary size	0.019 3	0.674	0.497 1	0.702

The algorithm chose KMCC (60 minutes, 2021 dataset).

4.7. Experimentation with Dictionary Size. In addition to the experiment conducted in the previous section, we have also experimented with the dictionary size of KAF algorithms. The result for this experiment is presented in Table 7. As visible, increasing the dictionary size decreases the performance of the system. Moreover, increasing the dictionary size also increased the execution time. It should be noted here that the execution time for predicting the next closing price for a single stock with dictionary size 500 is 0.675 seconds. This figure (0.675 seconds) clearly shows the applicability of the KAF class of algorithms in high-frequency trading, where latency is a key factor.

4.8. Important Note: Error Minimization and Profitability. From Tables 3 and 4, we can see that KRLS performed well in minimizing error. Moreover, the lowest error (MSE) that we get is in the order of 10^{-4}. It should be noted here that we got this error for the time window of one minute. In this regard, it is common sense that if we minimize the error, we can get close to the actual values, which is indeed true. However, considering the time window of one minute, there is an issue. In this interval, the fluctuation in the price is low. This means that minimizing error will not result in too much profit. In other words, the volatility in one minute is less. Hence, predictions are very close. However, the chances of taking a position and getting profit in a low volatile environment is also very less. Therefore, one has to maintain a balance between error minimization and profitability.

5. Conclusion

This paper introduces a framework to predict stock prices using KAF. We comprehensively analyzed the Indian Financial Sector, Nifty-50, and showed the predictive results of all 50 stocks in the main index. We experimented with ten different algorithms belonging to the KAF class of algorithms. Experimentation was performed on nine different windows starting at one minute and progressing to one day. This is the first time, to our knowledge, that numerous KAF algorithms have been implemented at such granular levels. The evidence offered in the Experimental Results section demonstrated the work's overall predictive capability. It was discovered that the KAF class of algorithms not only outperformed other algorithms in terms of error minimization but also had a very short execution time, underlining its usefulness in the field of high-frequency trading.

For future work, we would test the framework via the application of hyperparameter optimization. This would be beneficial because KAF algorithms must deal with a wide range of hyperparameter settings. We will also use several hyperparameter optimization strategies to improve the model's accuracy.

Data Availability

The datasets and all related materials are available for download from the following website: shorturl.at/lnvF2.

Conflicts of Interest

The authors declare that there are no conflicts of interest in this research.

References

[1] A. Agrawal, J. Gans, and A. Goldfarb, *Prediction Machines: The Simple Economics of Artificial Intelligence*, Harvard Business Press, Boston, MA, USA, 2018.

[2] P. Goodwin, D. Önkal, and M. Thomson, "Do forecasts expressed as prediction intervals improve production planning decisions?" *European Journal of Operational Research*, vol. 205, no. 1, pp. 195–201, 2010.

[3] Z. Karevan and J. A. K. Suykens, "Transductive lstm for time-series prediction: an application to weather forecasting," *Neural Networks*, vol. 125, pp. 1–9, 2020.

[4] A. Abraham, N. S. Philip, and P. Saratchandran, "Modeling chaotic behavior of stock indices using intelligent paradigms," 2004, https://arxiv.org/abs/cs/0405018.

[5] D. Berger, K. Pukthuanthong, and J. Jimmy Yang, "International diversification with frontier markets," *Journal of Financial Economics*, vol. 101, no. 1, pp. 227–242, 2011.

[6] R. Singh and S. Srivastava, "Stock prediction using deep learning," *Multimedia Tools and Applications*, vol. 76, no. 18, pp. 18569–18584, 2017.

[7] S. Mishra, T. Ahmed, and V. K. Mishra, "Close-price prediction using online kernel adaptive filtering," in *Proceedings of the 13th International Conference on Contemporary Computing*, pp. 217–222, Noida, India, August 2021.

[8] E. F. Fama, "Efficient capital markets: a review of theory and empirical work," *The Journal of Finance*, vol. 25, no. 2, pp. 383–417, 1970.

[9] 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.

[10] E. Guresen, G. Kayakutlu, and T. U. Daim, "Using artificial neural network models in stock market index prediction," *Expert Systems with Applications*, vol. 38, no. 8, pp. 10389–10397, 2011.

[11] X. Liang, H. Zhang, J. Xiao, and Y. Chen, "Improving option price forecasts with neural networks and support vector regressions," *Neurocomputing*, vol. 72, no. 13-15, pp. 3055–3065, 2009.

[12] L. K. C. Chan, J. Lakonishok, and B. Swaminathan, "Industry classifications and return comovement," *Financial Analysts Journal*, vol. 63, no. 6, pp. 56–70, 2007.

[13] J. Patel, S. Shah, P. Thakkar, and K. Kotecha, "Predicting stock market index using fusion of machine learning techniques," *Expert Systems with Applications*, vol. 42, no. 4, pp. 2162–2172, 2015.

[14] T. W. Sagala, M. S. Saputri, R. Mahendra, and I. Budi, "Stock price movement prediction using technical analysis and sentiment analysis," in *Proceedings of the 2020 2nd Asia Pacific Information Technology Conference*, pp. 123–127, Bali Island, Indonesia, January 2020.

[15] 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.

[16] A. Pimenta, C. A. L. Nametala, F. G. Guimarães, and E. G. Carrano, "An automated investing method for stock market based on multiobjective genetic programming," *Computational Economics*, vol. 52, no. 1, pp. 125–144, 2018.

[17] 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.

[18] T.-C. Huang, R. N. Zaeem, and K. S. Barber, "It is an equal failing to trust everybody and to trust nobody," *ACM Transactions on Internet Technology*, vol. 19, no. 4, pp. 1–20, 2019.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] W. Liu, J. C. Principe, and S. Haykin, *Kernel Adaptive Filtering: A Comprehensive Introduction*, Vol. 57, John Wiley & Sons, , New York, NY, USA, 2011.

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

[25] L. Yu, S. Wang, and K. K. Lai, "An online learning algorithm with adaptive forgetting factors for feedforward neural networks in financial time series forecasting," *Nonlinear Dynamics and Systems Theory*, vol. 7, no. 1, pp. 51–66, 2007.

[26] 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.

[27] T. Guo, Z. Xu, X. Yao, H. Chen, K. Aberer, and K. Funaya, "Robust online time series prediction with recurrent neural networks," in *Proceedings of the 2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pp. 816–825, IEEE, Montreal, Canada, October 2016.

[28] 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.

[29] F. E. H. Tay and L. Cao, "Application of support vector machines in financial time series forecasting," *Omega*, vol. 29, no. 4, pp. 309–317, 2001.

[30] P. C. S. Bezerra and P. H. M. Albuquerque, "Volatility forecasting via SVR-GARCH with mixture of Gaussian kernels," *Computational Management Science*, vol. 14, no. 2, pp. 179–196, 2017.

[31] L. O. Freitas, P. R. Henriques, and P. Novais, "Analysis of human activities and identification of uncertain situations in context-aware systems," *International Journal of Artificial Intelligence*, vol. 18, no. 2, 2020.

[32] R.-E. Precup, T.-A. Teban, A. Albu, A.-B. Borlea, I. A. Zamfirache, and E. M. Petriu, "Evolving fuzzy models for prosthetic hand myoelectric-based control," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 7, pp. 4625–4636, 2020.

[33] C. Pozna and R.-E. Precup, "Applications of signatures to expert systems modelling," *Acta Polytechnica Hungarica*, vol. 11, no. 2, pp. 21–39, 2014.

[34] U. L. Yuhana, N. Z. Fanani, E. M. Yuniarno, S. Rochimah, L. T. Koczy, and M. H. Purnomo, "Combining fuzzy signature and rough sets approach for predicting the minimum passing level of competency achievement," *International Journal of Artificial Intelligence*, vol. 18, pp. 237–249, 2020.

[35] E.-L. Hedrea, R.-E. Precup, R.-C. Roman, and E. M. Petriu, "Tensor product-based model transformation approach to tower crane systems modeling," *Asian Journal of Control*, vol. 23, pp. 1313–1323, 2021.

[36] Y.-S. Chen, C.-H. Cheng, and W.-L. Tsai, "Modeling fitting-function-based fuzzy time series patterns for evolving stock index forecasting," *Applied Intelligence*, vol. 41, no. 2, pp. 327–347, 2014.

[37] 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.

[38] J.-J. Wang, J.-Z. Wang, Z.-G. Zhang, and S.-P. Guo, "Stock index forecasting based on a hybrid model," *Omega*, vol. 40, no. 6, pp. 758–766, 2012.

[39] M. J. Mauboussin, "Revisiting market efficiency: the stock market as a complex adaptive system," *The Journal of Applied Corporate Finance*, vol. 14, no. 4, pp. 47–55, 2002.

[40] E. F. Fama, "Random walks in stock market prices," *Financial Analysts Journal*, vol. 51, no. 1, pp. 75–80, 1995.

[41] 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. 15274–15290, 2020.

[42] T. Singh, N. Saxena, M. Khurana, D. Singh, M. Abdalla, and H. Alshazly, "Data clustering using moth-flame optimization algorithm," *Sensors*, vol. 21, no. 12, p. 4086, 2021.

[43] M. Lippi, M. Bertini, and P. Frasconi, "Short-term traffic flow forecasting: an experimental comparison of time-series analysis and supervised learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 2, pp. 871–882, 2013.

[44] W. Long, Z. Lu, and L. Cui, "Deep learning-based feature engineering for stock price movement prediction," *Knowledge-Based Systems*, vol. 164, pp. 163–173, 2019.

[45] A. Nahil and A. Lyhyaoui, "Short-term stock price forecasting using kernel principal component analysis and support vector

machines: the case of casablanca stock exchange," *Procedia Computer Science*, vol. 127, pp. 161–169, 2018.

[46] 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.

[47] Z. Liu, C. K. Loo, and K. Pasupa, "Recurrent kernel online sequential extreme learning machine with kernel adaptive filter for time series prediction," in *Proceedings of the 2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1–7, IEEE, Honolulu, HI, USA, November 2017.

[48] 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.

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

[50] 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.

[51] 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, Australia, April 2015.

[52] S. Garcia-Vega, X.-J. Zeng, and J. Keane, "Stock price prediction using kernel adaptive filtering within a stock market interdependence approach," *SSRN*, 2018.

[53] S. Van Vaerenbergh and I. Santamaría, "A comparative study of kernel adaptive filtering algorithms," in *Proceedings of the 2013 IEEE Digital Signal Processing and Signal Processing Education Meeting (DSP/SPE)*, pp. 181–186, IEEE, Napa, CA, USA, August 2013.

[54] C. Richard, J. C. M. Bermudez, and P. Honeine, "Online prediction of time series data with kernels," *IEEE Transactions on Signal Processing*, vol. 57, no. 3, pp. 1058–1067, 2008.

[55] 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, August 2011.

[56] 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, p. 103327, 2020.

[57] X. Wu, H. Chen, J. Wang, L. Troiano, V. Loia, and H. Fujita, "Adaptive stock trading strategies with deep reinforcement learning methods," *Information Sciences*, vol. 538, pp. 142–158, 2020.

[58] W. Liu and J. C. Príncipe, "Kernel affine projection algorithms," *EURASIP Journal on Applied Signal Processing*, vol. 2008, Article ID 784292, 12 pages, 2008.

[59] 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.

[60] 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.

[61] 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.

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

[63] 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.