FINAL PROJECT: REGRESSION FINANCE

January 3, 2026

Franz Masatoshi Yuri, Oumouhani El Vilaly, Fredo Alejos Arrieta

CONTENTS

1 INTRODUCTION

Stock market forecasting is a notoriously challenging task for ML models due to the nonstationary nature of financial time series. Although solutions involving neural networks, SVM, genetic algorithms etc have been tested with in this application with mixed results, online kernel adaptive filtering (KAF) appears as an interesting alternative to traditional ML tools for time-series prediction.

The objective of this project is therefore to evaluate the performance of KAF techniques, specifically Kernel Least Mean Square (KLMS) and Kernel Affine Projection Algorithm (KAPA), on predicting the value of TITAN stocks in Nifty-50 (the Indian Stock Index). The results will then be compared against traditional methods such as AMRules, KNN, SGD and SGD Regressor based on MSE, MEA and DS.

2 THEORETICAL BACKGROUND

2.1 Kernel Adaptative Filtering (KAF)

The fundamental objective of KAF algorithms is to learn a non-linear input-output mapping $f : S \longrightarrow \mathbb{R}$ based on a sequence of samples $((s_1, d_1), (s_2, d_2), \ldots, (s_n, d_n))$, where $s_i \in S \subset \mathbb{R}^L$ is the system input vector at time i, and d_i is the desired response.

In order to do that, KAF algorithms rely on mapping the input data from the original input space S into a high-dimensional Reproducing Kernel Hilbert Space (RKHS), denoted as $\mathbb{F}$. This mapping is achieved via a feature map $\phi(s)$, which transforms the non-linear problem in the input space into a linear problem in the feature space.

A key advantage of this approach is that inner products in the high-dimensional feature space can be computed using a kernel function κ in the input space, avoiding computationally expensive operations in $\mathbb{F}$.

The kernel function is defined as:

$$\kappa\langle s, s'\rangle = \langle \phi(s), \phi(s')\rangle_{\mathbb{F}}$$

In this study we will restrict our analysis to the Gaussian kernel, formulated as:

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

where σ represents the kernel width.

The changes that emerge from this procedure can be easily perceived by comparing the traditional LMS algorithm with its kernel version, the Kernel Least Mean Square.

2.2 Kernel Least Mean Square (KLMS)

The transition from the linear LMS to the Kernel LMS (KLMS) is defined by the adaptation of the algorithm when applied to the high-dimensional feature space $\mathbb{F}$.

The standard linear LMS is characterized by a fixed-weight vector ω that minimizes the prediction error $e_i = d_i - \omega_{i-1}^T s_i$. Meanwhile, the stochastic gradient descent update rule for this vector is given by

$$\omega_i = \omega_{i-1} + \eta e_i s_i \tag{1}$$

where η represents the step size.

The KLMS algorithm, on the other hand, applies this identical logic to the transformed data sequence $\{(\phi(s_i), d_i)\}$, substituting the input s_i with the feature vector $\phi(s_i)$ and therefore producing the update rule in $\mathbb{F}$

$$\Omega_i = \Omega_{i-1} + \eta e_i \phi(s_i) \tag{2}$$

Unlike the linear LMS, the weight vector $\mathbf{\Omega}_i$ in KLMS cannot be stored explicitly due to the potentially infinite dimensionality of $\mathbb{F}$. However, based on the Representer Theorem, $\mathbf{\Omega}_i$ can be expressed as a linear combination of the transformed input samples processed so far. By applying the kernel trick, the system output can be computed without accessing the feature space coordinates directly.

The algorithm effectively assigns a new kernel unit for every new sample, weighted by the error scaled by the step size. The functional update rule is summarized as

$$f_i = f_{i-1} + \eta e_i \kappa\langle s_i, \cdot\rangle$$

with $f_0 = 0$.

Consequently, the predictive function at iteration i is represented as a growing sum of kernel functions centered at the previous input samples:

$$f_i(s) = \sum_{j=1}^{i} o_j(i)\kappa\langle s_j, s\rangle$$

where $o_j(i)$ are the coefficients and $\{s_j\}_{j=1}^i$ are the centers stored in the dictionary during the training process.

This structure highlights the non-parametric nature of KLMS, where the model complexity grows linearly with the number of training samples.

2.3 Kernel Affine Projection Algorithm (KAPA)

The Kernel Affine Projection Algorithm (KAPA) is introduced to enhance performance specifically in scenarios affected by gradient noise. While standard KAF methods might rely on instantaneous error, KAPA seeks to minimize the cost function based on the error over a sequence of inputs, formulated as

$$\min_{\Omega} |d - \Omega^T \phi(s)|^2 \tag{3}$$

To solve this, the algorithm replaces the traditional covariance and cross-covariance matrix-vector computations with a local approximation derived directly from the data using stochastic gradient descent. The resulting weight update rule is expressed as

$$\Omega_i = \Omega_{i-1} + \eta\psi(i)[d(i) - \psi(i)^T\Omega_{i-1}] \tag{4}$$

where $\psi(i)$ represents the matrix of feature vectors corresponding to the most recent observations

$$\psi(i) = [\phi(s_{i-K+1}), \ldots, \phi(s_i)] \tag{5}$$

and K denotes the observation and regressor parameter.

3 RESULTS ANALYSIS

For both algorithms we used the TITAN stock so we can have the same environment as in the paper and throughout algorithms.

3.1 Kernel Least Mean Square (KLMS)

After running the code corresponding to the Kernel Least Mean Square we obtained the following results.

In the case of the metrics, we summarized the results in the table 1. For the case of MAE measure, we obtained better results than the paper, while the paper was better for the cases of MSE and DS. This pattern repeated in both types of mid price (high/low and open/close).

Source	**Window**	**MSE**	**MAE**	**DS (%)**
Mid-Price: (High+Low)/2				
KLMS (Ours)	1 day	0.0925	**0.2361**	43.48
KLMS (Ours)	30 min	0.0330	**0.1222**	47.68
KLMS (Paper)	30 min	**0.0053**	0.3595	**55.58**
Mid-Price: (Open+Close)/2				
KLMS (Ours)	1 day	0.0725	**0.2154**	52.17
KLMS (Ours)	30 min	0.0251	**0.0916**	51.90
KLMS (Paper)	30 min	**0.0048**	0.3168	**65.47**

Table 1: KLMS performance

Between the plots we obtained the figure 1, which have the performance of the KLMS algorithm using windows of 30 minutes, and figure 2, which have the error obtained for the same KLMS algorithm and algorithms. As we can see there are some moments of unpredictability, which is normal in the stocks value, but after a while the model seems to be able to understand the new trend.

Figure 1: KLMS Performance using windows of 30 minutes

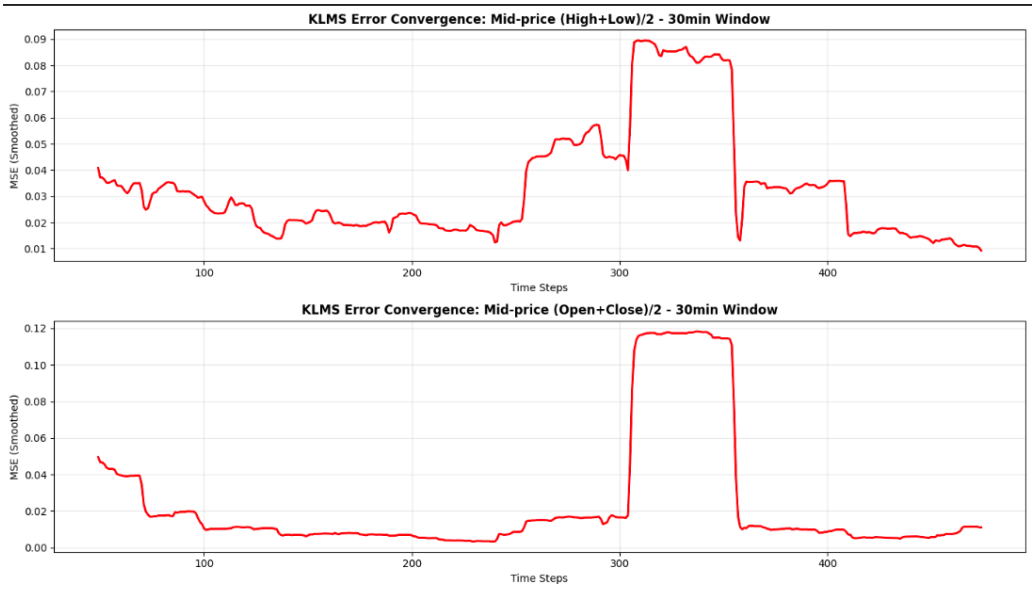

Figure 2: KLMS Error Convergence for windows of 30 minutes

3.2 Kernel Affine Projection Algorithm (KAPA)

After running the code corresponding to the Kernel Affine Projection Algorithm. We obtained the following results.

After running the code corresponding to the Kernel Least Mean Square we obtained the following results.

In the case of the metrics, we summarized the results in the table 2. For the case of MAE measure, we obtained better results than the paper, while the paper was better for the cases of MSE and DS. This pattern repeated in both types of mid price (high/low and open/close). This is nearly the same results we obtained with the previous algorithm.

Source	Window	MSE	MAE	DS (%)
Mid-Price: (High+Low)/2				
KAPA (Ours)	1 day	0.0925	**0.2361**	43.48
KAPA (Paper)	1 day	**0.0306**	1.4129	**53.78**
KAPA (Ours)	30 min	0.0192	**0.0940**	39.87
KAPA (Paper)	30 min	**0.0053**	0.3595	**55.58**
Mid-Price: (Open+Close)/2				
KAPA (Ours)	1 day	0.0725	**0.2155**	52.17
KAPA (Paper)	1 day	**0.0324**	1.2989	**59.70**
KAPA (Ours)	30 min	0.0149	**0.0732**	48.52
KAPA (Paper)	30 min	**0.0048**	0.3168	**65.47**

Table 2: KAPA performance

Between the plots we obtained, we have the figure 3, which shows the performance of the KAPA predicition using 30 minutes windows, and the figure 4, which shows the error obtained from figure 3. As we can see there are some similar patterns to the previous algorithm performance. This algorithm seems to also be able to cope to new trends after some time.

Figure 3: KAPA Performance using windows of 30 minutes

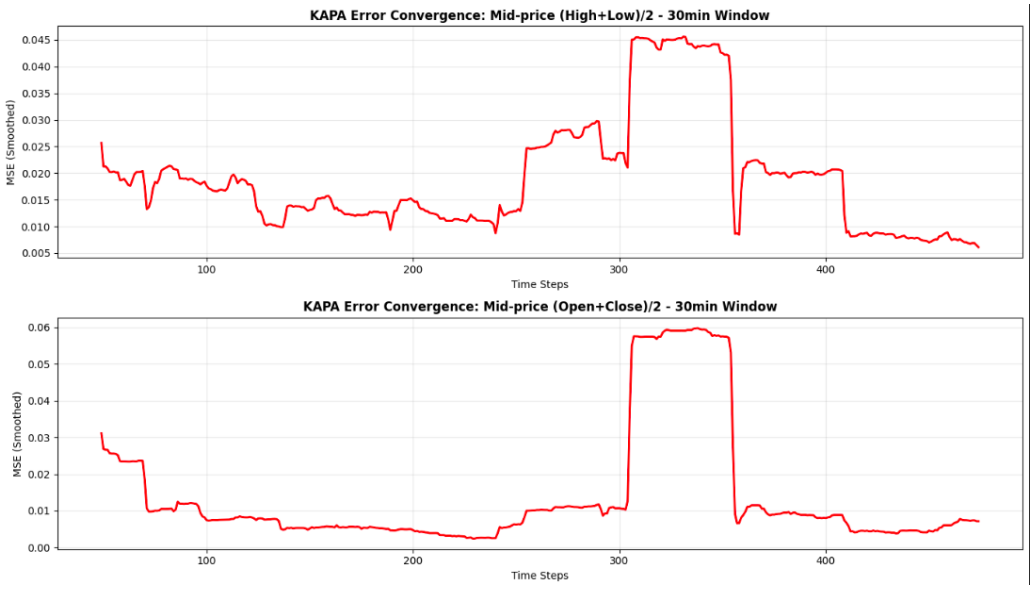

Figure 4: KAPA Error Convergence for windows of 30 minutes

4 CONCLUSIONS

- We successfully replicated and implemented the algorithms of KAPA and KLMS for Kernel Adaptive Filtering, these were applied to recen TITAN stock data.
- Models trained using 30 minutes outperformed those of the paper in MAE metric, while it didn't in MSE and DS.
- An important detail to mention is that the low prediction error, such as MAE and MSE, does not guarantee good directional accuracy (DS).