Stock returns prediction using kernel adaptive filtering within a stock market interdependence approach

DOI:
10.1016/j.eswa.2020.113668

Document Version
Accepted author manuscript

Link to publication record in Manchester Research Explorer

Citation for published version (APA):

Published in:
Expert Systems with Applications

Citing this paper
Please note that where the full-text provided on Manchester Research Explorer is the Author Accepted Manuscript or Proof version this may differ from the final Published version. If citing, it is advised that you check and use the publisher's definitive version.

General rights
Copyright and moral rights for the publications made accessible in the Research Explorer are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

Takedown policy
If you believe that this document breaches copyright please refer to the University of Manchester’s Takedown Procedures [http://man.ac.uk/04Y6Bo] or contact uml.scholarlycommunications@manchester.ac.uk providing relevant details, so we can investigate your claim.
Stock returns prediction using kernel adaptive filtering within a stock market interdependence approach

Sergio Garcia-Vega\textsuperscript{a,*}, Xiao-Jun Zeng\textsuperscript{a}, John Keane\textsuperscript{a}

\textsuperscript{a}School of Computer Science, University of Manchester, Kilburn Building, Oxford Road, Manchester, M13 9PL, United Kingdom

Abstract

Stock returns are continuously generated by different data sources and depend on various factors such as financial policies and national economic growths. Stock returns prediction, unlike traditional regression, requires consideration of both the sequential and interdependent nature of financial time-series. This work uses a two-stage approach, using kernel adaptive filtering (KAF) within a stock market interdependence approach to sequentially predict stock returns. Thus, unlike traditional KAF formulations, prediction uses not only their local models but also the individual local models learned from other stocks, enhancing prediction accuracy. The enhanced KAF plus market interdependence framework has been tested on 24 different stocks from major economies. The enhanced approach obtains higher sharpe ratio when compared with KAF-based methods, long short-term memory, and autoregressive-based models.

Keywords: Stock returns prediction; Sequential learning; Interdependence between markets; Kernel adaptive filtering.

1. Introduction

Stock returns are continuously generated by different and separate data sources, such as banks, corporations, and individuals. Thus, unlike traditional...
classification and regression problems, predicting these time-series requires consideration of: i) a continuous sequence of data records (Cui et al., 2016); ii) the interdependency between stock markets (Rejeb & Arfaoui, 2016). The stock returns are characterized by nonlinearities and high noise, which imposes additional challenges to obtaining reliable prediction (Ballings et al., 2015).

We introduce a two-phase framework for stock returns prediction using sequential learning within a stock market interdependence approach. Thus, the underlying models of each stock are learned separately using a kernel-based adaptive filter that encodes different patterns of the input space. Further, stock returns are predicted using not only their local models, but also the individual local models learned from other stocks, providing a natural way to incorporate these interdependencies. The proposed framework is a distributed learning paradigm rather than a centralized one in the sense that individual prediction models are learned based solely on a local data store, thus avoiding expensive and time-consuming data transportation into an integrated, central data store. Such a distributed learning paradigm is especially critical for big data analysis and real-time learning (Scardapane et al. 2016; Sohangir et al. 2018).

The framework is validated on 24 different stocks from three major economies, i.e., Germany, United States, and United Kingdom. Simulation obtains low values of mean absolute error (MAE) and mean square error (MSE) when compared with kernel-based adaptive filter, long short-term memory, and autoregressive-based methods. In addition, local models learned from stocks in the UK and the US were used to predict stocks traded in Germany, maximizing returns and reducing volatility.

The remainder of the paper is structured as follows: Section 2 shows the relevant background and related work; Section 3 introduces the proposed framework; the experimental settings are described in Section 4; Section 5 presents simulation results and analysis; and Section 6 concludes.
2. Background and Related Work

2.1. Sequential Learning

Several prediction methods that consider the sequential and interdependent nature of financial time series have been proposed. Sequential learning is usually addressed using the following methods: i) statistical models, such as autoregressive integrated moving average (Lee & Tong, 2011; Ramos et al., 2015). However, such linear models may not be well suited for characterizing the stochastic nature and uncertain dynamics of financial time series (Khashei & Hajirahimi, 2018; Suhermi et al., 2018); ii) feedforward neural networks (FFNNs) (Goodfellow et al., 2016), where sequential data is modeled by adding a set of delays to the input, e.g., time-delay NNs (Huang et al., 2017), that deals with the shortest path problem by means of time-dependent neurons; iii) recurrent NNs (Pascanu et al., 2013), that model sequence structure with recurrent lateral connections and process the data sequentially one record at a time, e.g., long short-term memory (LSTM) (Fischer & Krauss, 2018); iv) kernel adaptive filters (KAFs), that combine the universal approximation property of NNs and the convex optimization of linear adaptive filters, e.g., kernel least-mean-square (KLMS) (Liu et al., 2008).

The prediction of financial time series has been addressed with a variety of NN approaches (Rather et al., 2015; Liu et al., 2015; Doucoure et al., 2016; Hosaka, 2019). However, these strategies may fail, as they can get stuck in local minima during the training stage (Principe, 2010). In addition, these methods require all the training data to be fully prepared in advance, restricting their application in sequence prediction (Gu et al., 2014). In practice, as NN models are retrained at regular intervals they require both significant computing and storage resources (Cui et al., 2016). Note that non-parametric kernel approaches have proven useful in identifying non-linear systems (Orabona et al., 2009; Zhao et al., 2011; Tang et al., 2017), showing that their convex optimization helps to reduce the computational complexity in sequential learning environments (Liu et al., 2011). The KAFs-based approaches can start learning the model without
having the entire training set in advance, as their learning scheme is a combination of memory-based learning and error-correction, meaning that the model is updated sequentially in real-time while predictions are obtained. Hence, in this work, we use a KAF-based sequential learning method to model nonlinearities in financial time series.

2.2. Interdependency between stock markets

In addition to the sequential nature of financial time series, stock markets are themselves highly complex systems depending on various factors such as financial policies, national economic growths and sector performance (Zhang et al., 2019). It has became clear that all agents involved in a given stock market may exhibit interconnections and correlations, representing important internal forces of the market (Collins & Biekpe, 2003; Jizba et al., 2012) - that is, the movement of a stock market in a country is likely to be affected by movement of other stocks in both that country and in other regions (Masih & Masih, 2001). The following strategies have been proposed to identify and quantify interactions on this type of complex system (Greenblatt et al., 2012): i) space-time, such as covariance (Wang & Ye, 2016; De Ketelaere et al., 2018), correlation (Kenett et al., 2015), Granger causality (Papana et al., 2017), Shannon entropy (Sulthan et al., 2016), mutual information (Wang & Hui, 2017), and Renyi entropy (Brody et al., 2007); ii) space-frequency and space-time-frequency, such as Fourier transform (Fang & Chang, 2017; Saia et al., 2017), coherence (Vacha & Barunik, 2012), phase synchronization (Radhakrishman et al., 2016), directed transfer function ( Kamiński et al., 2001), wavelet transform (Joo & Kim, 2015; Saia, 2017), and cross-time frequency measures (Loh, 2013). The previous works study how the price of one stock is influenced by the economic factors of other markets. However, their models do not consider changes in network structure over time, meaning that the conditions for which the models were optimized may disappear (Olsen et al., 2018). Hence, how to incorporate these interdependencies into an analytical model, such as sequential learning, to predict stock returns in real-time remains an open issue.
3. Proposed sequential prediction and stock market interdependence framework

We develop a two-stage framework for stock returns prediction: (1) sequential learning, where the underlying models of each stock are learned separately using KAFs (Section 3.1); (2) interdependence between stocks, where local models learned from different stock markets are used to improve prediction (Section 3.2).

3.1. Sequential learning based on kernel adaptive filtering

Given a set of training data $\mathcal{T} = \{\mathbf{u}_t, y_t : t \in [1, N]\}$, where $\mathbf{u}_t \in \mathbb{R}^M$ is an input vector and $y_t \in \mathbb{R}$ is the desired output (see Figure 1). The task is to infer the underlying function $y = f(\mathbf{u})$ from the given data $\mathcal{T}$ and, for a new input vector $\mathbf{u}' \in \mathbb{R}^M$, predict the value of a new observation $y' \in \mathbb{R}$.

In practice, a KAF sequentially estimates $f$ by using the current input-output pair $\{\mathbf{u}_t, y_t\}$ and updating the previous estimate $f_{t-1}$ as follows (Liu et al., 2008):

$$
\begin{align*}
    f_0 &= 0 \\
    e_t &= y_t - f_{t-1}(\mathbf{u}_t) \\
    f_t &= f_{t-1} + \eta e_t \kappa_\sigma(\mathbf{u}_t, \cdot)
\end{align*}
$$

(1)

being $\eta \in \mathbb{R}^+$ the step-size, $f_t$ the learned mapping, $e_t \in \mathbb{R}$ the prediction error, while $\kappa_\sigma(\cdot, \cdot) \in \mathbb{R}^+$ is a Mercer kernel with a bandwidth $\sigma \in \mathbb{R}^+$ that controls the mapping smoothness (Schölkopf & Smola, 2001). Note that Equation (1) creates a kernel unit for every new sample, where $\mathbf{u}_t$ is the center and $\eta e_t$ its coefficient, posing additional issues for continuous adaptation scenarios. A challenge is to curb the network growth by either eliminating redundant information or minimizing information loss - that is, only using input data with high information content as the new centers.

We propose to reduce the network size by partitioning the centers into distinct regions that encode different patterns of the input space. Thus, as it has shown
stable performance in non-stationary environments, these patterns are identified using a previously proposed change point detection method (Yamanishi & Takeuchi, 2002). This method, at each iteration \( t \), determines whether a change in the distribution has occurred within the sequence \( y_1, \ldots, y_t \). This is done by measuring how large the probability density function \( p_t \) has moved from \( p_{t-1} \) after learning from \( y_t \). Particularly, it is stated that a change point has taken place at iteration \( t \) when the following inequality holds:

\[
\epsilon(y_t, p_{t-1}) \geq \delta
\]  

(2)

where \( \epsilon(y_t, p_{t-1}) = -\log p_{t-1}(y_t) \) denotes a prediction loss for \( y_t \) relative to a probability density function \( p_{t-1} \), while \( \delta \in \mathbb{R}^+ \) is a predefined threshold. Thus, when a change-point is detected within the sequence \( y_1, \ldots, y_t \), we form a new set of centers or dictionary. Additionally, to avoid large discontinuities in learn-
ing (Li & Príncipe, 2017), all the centers and coefficients of the closest dictionary are copied or transferred to the newly formed dictionaries as follows:

(i) $\epsilon(y_t, p_{t-1}) \geq \delta$: This indicates a change in data distribution; hence, the following three dictionaries are formed,

- $\mathcal{C}_{|\mathcal{C}|+1} = \{C^*_i, u_t\}$, with $\mathcal{C} = \{C_i : i \in [1, |\mathcal{C}|]\}$;
- $\mathcal{Y}_{|\mathcal{Y}|+1} = \{Y^*_i, y_t\}$, with $\mathcal{Y} = \{Y_i : i \in [1, |\mathcal{Y}|]\}$;
- $\mathcal{W}_{|\mathcal{W}|+1} = \{W^*_i, \eta_e_t\}$, with $\mathcal{W} = \{W_i : i \in [1, |\mathcal{W}|]\}$;

where $|\mathcal{C}|=|\mathcal{Y}|=|\mathcal{W}|$ denotes the number of elements in each dictionary. In addition, $C^*_i = \{u_j : j \in [1, L]\}$, $Y^*_i = \{y_j : j \in [1, L]\}$, and $W^*_i = \{\eta_e_j : j \in [1, L]\}$ are the closest dictionaries to $y_{t-1}$. Note, $C^*_i$, $Y^*_i$, and $W^*_i$ are found using $y_{t-1}$ rather than $y_t$. This is because, during real-time prediction tasks, only the input vectors $u_t$ are known and the desired output $y_t$ is the value to be predicted. As seen in Figure 4, $y_{t-1}$ is always the last element of the input vector $u_t$. We find the closest dictionary $i^*$ using the Kullback-Leibler divergence as:

$$i^* = \arg\min_{\forall i} p_{t-1}^{Y_i} \ln \left( \frac{p_{t-1}^{Y_i}}{p_t^{Y_i}} \right)$$

where $p_{t-1}^{Y_i}$ and $p_t^{Y_i}$ are the probability density functions of $Y_i$ before and after learning $y_{t-1}$, respectively. The primary rationale behind the suggested strategy in Equation (3) is to quantify the information content that $y_{t-1}$ will provide to each dictionary $Y_i$. That is, when $y_{t-1}$ does not provide high information content to a dictionary $Y_i$, the Kullback-Leibler divergence will tend to zero, meaning that the two distributions $p_{t-1}^{Y_i}$ and $p_{t}^{Y_i}$ are identical. Thus, the dictionary $Y_i$ that minimizes Equation (3) will be the closest dictionary to $y_{t-1}$.

(ii) $\epsilon(y_t, p_{t-1}) < \delta$: This means that the data distribution has not changed and, therefore, it is unnecessary to divide the centers into a new region.

In addition, to curb the growth of the radial-basis-function structure, we incorporate an online vector quantization technique as follows,
\[ \min_{Y_{i}} \min_{p_{Y_{i}}^{i-1} \leq \frac{p_{Y_{i}}}{p_{Y_{i}}^{i-1}}} \ln \left( \frac{p_{Y_{i}}^{i-1}}{p_{i}^{Y_{i}}} \right) \leq \varepsilon: \text{ The dictionary sizes remain the same and only the closest coefficient to } u_{i} \text{ is updated using the following expression,} \]

\[ W_{i}^{j_{*}} = W_{i}^{j_{*}} + \eta e_{t}, \quad (4) \]

where the closest coefficient \( j_{*} \) is computed as follows,

\[ j_{*} = \arg \min_{\forall j} \left\| u_{t} - C_{i}^{(j)} \right\|, \quad (5) \]

being \( \| \cdot \| \) the \( \ell_{2} \) norm and \( \varepsilon \in \mathbb{R}^{+} \) a predefined threshold. The previously imposed restraint aims to assign a new center \( u_{t} \) into the dictionary \( C_{i}^{j_{*}} \) only when \( y_{t-1} \) provides high information content to the dictionary \( Y_{i_{*}} \).

\[ \min_{Y_{i}} \min_{p_{Y_{i}}^{i-1} \leq \frac{p_{Y_{i}}}{p_{Y_{i}}^{i-1}}} \ln \left( \frac{p_{Y_{i}}^{i-1}}{p_{i}^{Y_{i}}} \right) > \varepsilon: \text{ The sample } u_{t} \text{ is assigned as a new center to the closest dictionary } C_{i}^{*}, \text{ the previous desired output } y_{t-1} \text{ is stored in the dictionary } Y_{i_{*}}, \text{ while the set of coefficients } W_{i}^{*} \text{ is updated using } e_{t}, \text{ i.e., } C_{i}^{*} = \{C_{i}^{*}, u_{t}\}, Y_{i}^{*} = \{Y_{i}^{*}, y_{t-1}\}, W_{i}^{*} = \{W_{i}^{*}, \eta e_{t}\}. \]

The above quantization technique has similarities to the method in Chen et al. (2012). The key difference between the two strategies is that our proposal is not based on the distance measure in the input space. Rather, we use the data distribution as the criterion to update the network. This enhances utilization efficiency of the closest center, which may yield better prediction accuracy and in some circumstances a more compact network. The proposed strategy for sequential learning, when applied to a single stock, is summarized in Algorithm 1.
Algorithm 1: Strategy for sequential learning.

\textbf{input}: \mathcal{T} - training data

1. Parameter setting: \( \eta \)- step-size, \( \sigma \)- kernel parameter, \( \delta \)- change point threshold, \( \varepsilon \)- quantization threshold

2. Initial dictionaries: \( C_1 = \{ u_1 \}, \ Y_1 = y_0, \ W_1 = \eta y_1 \)

3. Sets of initial dictionaries: \( C = \{ C_1 \}, \ Y = \{ Y_1 \}, \ W = \{ W_1 \} \)

4. Computation:

5. \textbf{while} \( \{ u_t, y_t \} \) available \textbf{do}

6. Select the closest dictionary: \( i^* = \arg \min \forall i \ln \left( \frac{p_{r,i}^Y}{p_{r,i}^X} \right) \)

7. Compute the filter output: \( \hat{y}_t = \sum_{j=1}^L W_j^{(j)} \kappa(\xi_{i^*}^j, u_t) \)

8. Compute the error: \( e_t = y_t - \hat{y}_t \)

9. \textbf{if} \( \varepsilon(y_t, p_t - 1) \geq \delta \) \textbf{then}

10. Form new dictionaries:

11. \( C_{|C|+1} = \{ C_{i^*}, u_t \}, \ Y_{|Y|+1} = \{ Y_{i^*}, y_{t-1} \}, \ W_{|W|+1} = \{ W_{i^*}, \eta e_t \} \)

12. Update set of dictionaries:

13. \( C = \{ C, C_{|C|+1} \}, \ Y = \{ Y, Y_{|Y|+1} \}, \ W = \{ W, W_{|W|+1} \} \)

14. \textbf{else}

15. \textbf{if} \( \min \forall i \ln \left( \frac{p_{r,i}^Y}{p_{r,i}^X} \right) \leq \varepsilon \) \textbf{then}

16. Select the closest center: \( j^* = \arg \min \forall j \| u_t - C_{i^*}^j \| \)

17. Update coefficient of closest center: \( W_{j^*} = W_{j^*} + \eta e_t \)

18. \textbf{else}

19. Assign a new center: \( C_{i^*} = \{ C_{i^*}, u_t \} \)

20. Assign a new desired output: \( Y_{i^*} = \{ Y_{i^*}, y_{t-1} \} \)

21. Assign a new coefficient: \( W_{i^*} = \{ W_{i^*}, \eta e_t \} \)

22. \textbf{output}: \( C, \ Y, \ W \)

3.2. Stock returns prediction within a stock market interdependence approach

With the aim of enhancing stock returns prediction, we consider interdependencies between stock markets. More formally, let \( \mathcal{D} = \{ \mathcal{T}_r : r \in [1, S] \} \) be the set of training samples of \( S \) stocks, where \( \mathcal{T}_r = \{ u_{t,r}, y_{t,r}, t \in [1, N] \} \). The underlying models of each \( \mathcal{T}_r \), as seen in Figure 2 are learned separately using Algo-
algorithm giving three sets of dictionaries per stock, i.e., $C_r = \{C_{i,r} : i \in [1, |C_r|]\}$, $Y_r = \{Y_{i,r} : i \in [1, |Y_r|]\}$, and $W_r = \{W_{i,r} : i \in [1, |W_r|]\}$.

Figure 2: Sequential learning within a stock market interdependence approach.

Then, when a new input vector $u' \in \mathbb{R}^M$ arrives, the task is to predict a value $y' \in \mathbb{R}$. Thus, the first step is to find the closest dictionary $i^*$, as in Section 3.1, using the following expression:

$$i^* = \arg \min_{i,r} p_{Y_{i,r}} \ln \left( \frac{p_{Y_{i,r}}}{p_{Y_{i,r}}} \right),$$

(6)

from Equation (6), it can be seen that several stocks are considered in the selection of the closest dictionaries $C_r = \{u_j : j \in [1, L]\}$ and $W_r = \{\eta_j : j \in [1, L]\}$. In practice, these dictionaries are used to predict $y'$ as follows:

$$y' = \sum_{j=1}^{L} W_{i^*,j} \kappa_{\sigma} \left( C_{i^*,j}^{(j)}, u_t \right)$$

(7)

Note that, when a new sample $u'$ comes from the $r$-th stock, its prediction is usually calculated using the model learned on that stock. Here, we predict $u'$ using not only the local model but also the individual local models learned from other stocks (see Equations (6) and (7)). This strategy has similarities to previously proposed methods such as ensemble learning (Dietterich et al., 2002; Krawczyk et al., 2017) and forecast combination (Newbold & Harvey, 2002; Baumeister & Kilian, 2015). The ensemble learning framework is constructed in two steps (Zhou, 2015): 1) a number of base learners are produced, which can be generated in a parallel or sequentially; 2) the base learners are combined using majority voting for classification or weighted averaging for regression.
However, the combination of multiple classifiers does not always outperform the best individual classifier (Polikar, 2009). In addition, better results may be obtained when some base learners are selected instead of ensemble them (Zhou et al., 2002). In contrast to ensemble learning methods, our approach does not combine the base learners; rather, here, the prediction tasks are performed only by the best learner. This better utilises the internal forces of the market, provides a natural way to incorporate interdependencies between stock markets, and, in some circumstances, may enhance accuracy in real-time prediction tasks (Zhou & Tang, 2003). Finally, the proposed real-time prediction framework for stock returns is shown in Algorithm 2.

**Algorithm 2:** Predicting stock returns within a stock market interdependence approach.

```
input : \( u', C_1, \ldots, C_S; Y_1, \ldots, Y_S; W_1, \ldots, W_S \)
1 Parameter setting: \( \sigma \)-kernel parameter
2 Computation:
3 Select the closest dictionary
4 \( i^* = \arg \min_{i \in \mathcal{I}} \text{ln} \left( \frac{p_{t-1}^{Y_i,r}}{p_t^{Y_i,r}} \right) \)
5 Compute the output of the filter
6 \( y' = \sum_{j=1}^{L} W_j^{(j)} \kappa_\sigma \left( C_j^{(j)}, u_t \right) \)
output: \( y' \)
```

4. Experimental Design

The aim is to use the last ten stock returns to predict the current day price change. The learned filter, as in Liu et al. (2011), is used to compute the performance values on each test set. We validate the proposed framework for stock returns prediction using the following performance measures:

- **Mean Absolute Error (MAE),** a negatively-oriented score that measures the average of absolute errors - lower MAE values indicate better prediction performance (Baek & Kim, 2018);
- **Mean Squared Error (MSE)**, measures the average of the squares of the errors, meaning that values closer to zero are better (Kiran et al., 2020);

- **Sharpe Ratio (SR)**, quantifies the average return earned in excess of the risk-free rate per unit of volatility or total risk (Wang et al., 2020). Here, as suggested in Almahdi & Yang (2019), the Sharpe Ratio does not consider any risk-free rate.

The above metrics have been widely used to measure models’ predictive power and their trading performance (France & Ghose, 2019; Portugal et al., 2018; Kalayci et al., 2019). The first two performance measures are regression-oriented metrics, while the last one is considered the industry standard for measuring risk-adjusted return (Jalota et al., 2017). In addition, for comparison purposes, the proposed framework is contrasted with the following online prediction methods:

1. Long Short-Term Memory (LSTM), representing the state-of-the-art RNN model for sequence learning tasks (Nweke et al., 2018). Here, as it has shown competitive performance, the implementation known as vanilla LSTM is used (Greff et al., 2017);

2. Nearest Instance Centroid-Estimation (NICE) (Li & Príncipe, 2017), a recently proposed method that outperforms traditional KAF-based algorithms in prediction of chaotic time-series;

3. Quantized Kernel Least-Mean-Square (QKLMS) (Chen et al., 2012), a well-known adaptive filtering method that uses an online vector quantization strategy;

4. Vector Autoregression (VAR) (Jang, 2020), a forecasting method used to identify relationships among multiple time-series;

5. Vector Error Correction Model (VECM) (Liang & Schienle, 2019), which is considered the standard tool to handle multivariate non-stationary time-series.
The Python implementation of the above methods can be downloaded from

4.1. Dataset

The daily closing prices data used to calculate the returns are collected from
Yahoo Finance\footnote{https://finance.yahoo.com/}. Here, as suggested in Siikanen et al.\footnote{https://segarciave.github.io/stock_returns_prediction}, we calculate daily
log returns using the adjusted closing prices of each day. The publicly available
dataset can be downloaded using Python libraries such as yfinance (please visit
the following GitHub page for more details https://segarciave.github.io/)
stock_returns_prediction). Testing has been carried out on 24 different
stocks from three major economies over 12 years (see Table 1). The considered
training set ranges from January 17, 2006, to November 30, 2016, while the
test set covers January 3, 2017, to February 28, 2018. In the simulations, as
suggested in Chen\footnote{https://www.tensorflow.org/}, the dataset is standardized so all stocks have a mean
estimation of zero and a standard deviation of one.

4.2. Parameter settings

Table 2 summarizes the set-up of compared methods in the tested stocks.
The parameters were heuristically adjusted to provide the best possible accuracy
in this dataset. In particular, to ensure consistency in the results, both $\eta$ and $\sigma$
remain the same for all KAF methods. Thus, as long as this condition is met,
different $\eta$ and $\sigma$ values from those shown in Table 2 will not offer an advantage
to any particular algorithm.

We train the LSTM method using a single hidden layer with 20 neurons,
as suggested in Cui et al.\footnote{https://www.tensorflow.org/} (2016). The LSTM activation function, as suggested
in Tian et al.\footnote{https://www.tensorflow.org/} (2018), is the sigmoid, while the optimization is performed by
Adam algorithm with MSE as the loss function Kingma & Ba\footnote{https://www.tensorflow.org/} (2014). The
LSTM method was implemented using TensorFlow (version 1.4.0)\footnote{https://www.tensorflow.org/} and Keras
The free parameters of autoregressive-based methods (VAR and VECM) were selected using heuristic approaches and following strategies proposed in the literature \cite{Lütkepohl2013, Kuo2016}. These parameters were chosen to achieve the best MSE on each tested data set.

The performance of our proposal is sensitive to the selection of $\delta$ and $\varepsilon$; however, values for these parameters can be selected as follows: i) $\delta$-threshold, based on our experimentation, an appropriate value is in the interval $[5, 15]$; ii) $\varepsilon$-quantization value, where a value of 0.0001 has shown stable performance on all tested stocks.

5. Simulation Results and Analysis

The simulation results for the compared methods are shown in Tables 3 to 5, where the last row displays the average performance of each algorithm. The best overall method is in bold notation and marked with an asterisk. Table 3

\begin{table}[h]
\centering
\begin{tabular}{ll}
\hline
Market & Ticker & Stock \\
\hline
DE & ADS & Adidas AG \\
    & ALV & Allianz SE \\
    & DPW & Deutsche Post AG \\
    & DTE & Deutsche Telekom AG \\
    & HEI & Heidelberger Cement AG \\
    & LIN & Linde AG \\
    & MRK & Merck KGaA \\
    & SAP & SAP AG \\
\hline
UK & ADM & Admiral Group PLC \\
    & AHT & Ashtead Group PLC \\
    & BA & BAE Systems PLC \\
    & BP & BP PLC \\
    & CCL & Carnival PLC \\
    & IAG & International Consolidated Airlines Group \\
    & SKY & SKY PLC \\
    & VOD & Vodafone Group PLC \\
\hline
US & AAL & American Airlines Group Inc \\
    & AAPL & Apple Inc \\
    & AMZN & Amazon Inc \\
    & C & Citigroup Inc \\
    & GOOGL & Alphabet In-CL A \\
    & MSFT & Microsoft Corp \\
    & SPY & SPDR S&P 500 Etf \\
    & T & AT&T \\
\hline
\end{tabular}
\end{table}

\footnote{https://keras.io/}
Table 2: Parameter setting of compared methods. $M$-input vector size, $\eta$-step size, $\sigma$-bandwidth, $\lambda$-quantization value, $\beta$-centroid distance, $\delta$-threshold, $\varepsilon$-quantization value, $\mathcal{L}$-layers, $N$-neurons per layer, $G$-maximum number of lags, $A$-number of lagged differences in the model, $I$-cointegration rank.

<table>
<thead>
<tr>
<th>Method</th>
<th>Parameter</th>
<th>$M$</th>
<th>$\eta$</th>
<th>$\sigma$</th>
<th>$\lambda$</th>
<th>$\beta$</th>
<th>$\delta$</th>
<th>$\varepsilon$</th>
<th>$\mathcal{L}$</th>
<th>$N$</th>
<th>$G$</th>
<th>$A$</th>
<th>$I$</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM</td>
<td></td>
<td>10</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>20</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>NICE</td>
<td></td>
<td>10</td>
<td>0.05</td>
<td>0.5</td>
<td>0.06</td>
<td>2$\sigma$</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Proposal</td>
<td></td>
<td>10</td>
<td>0.05</td>
<td>0.5</td>
<td>0.06</td>
<td>10</td>
<td>0.0001</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>QKLMS</td>
<td></td>
<td>10</td>
<td>0.05</td>
<td>0.5</td>
<td>0.4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>VAR</td>
<td></td>
<td>10</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>15</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>VECM</td>
<td></td>
<td>10</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>

shows the MAE values of each method in the considered stocks. The LSTM method outperforms the other algorithms, converging to the lowest average MAE value. This means that an error no greater than 0.012 can be expected during the prediction task on average. Although LSTM shows the best performance, the compared methods also converge to relatively low values of MAE. Note, a method that minimises MAE will lead to forecasts of the median (Chai & Draxler, 2014), meaning that this scale-dependent metric may be unable to quantify the prediction of abrupt changes in stock returns.

Table 4 summarizes the MSE prediction performance, where lower values are better. The compared methods show similar MSE values, suggesting a relatively good prediction performance on all considered stocks under this metric. However, NN-based methods such as LSTM requires significant computing resources, as they need to be retrained regularly in sequence prediction tasks (Chai et al., 2016). In practice, this means that several epochs have to be re-performed each time a new sample arrives in the system, which allows finding the best possible performance when the training set is updated. The learning scheme of our proposal, unlike LSTM and autoregressive-based methods, does not require the entire training set in advance to start learning the model. In contrast, the model is updated sequentially while predictions are obtained at the same time, providing an alternative to sequence prediction tasks. Additionally, the proposed interdependence strategy allows predicting each stock using not only the local model but also the models learned from other stock markets, supporting the
Table 3: Testing MAE at final iteration in stock returns prediction.

<table>
<thead>
<tr>
<th>Stock</th>
<th>Method 1</th>
<th>Method 2</th>
<th>Method 3</th>
<th>Method 4</th>
<th>Method 5</th>
<th>Method 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADS</td>
<td>0.0109</td>
<td>0.0109</td>
<td>0.0137</td>
<td>0.0109</td>
<td>0.0115</td>
<td>0.0136</td>
</tr>
<tr>
<td>ALV</td>
<td>0.01</td>
<td>0.0101</td>
<td>0.0122</td>
<td>0.01</td>
<td>0.0101</td>
<td>0.0122</td>
</tr>
<tr>
<td>DPW</td>
<td>0.0611</td>
<td>0.0608</td>
<td>0.0645</td>
<td>0.0609</td>
<td>0.0601</td>
<td>0.0711</td>
</tr>
<tr>
<td>DTE</td>
<td>0.0063</td>
<td>0.0063</td>
<td>0.0142</td>
<td>0.0063</td>
<td>0.0065</td>
<td>0.0763</td>
</tr>
<tr>
<td>HEI</td>
<td>0.0099</td>
<td>0.0097</td>
<td>0.0161</td>
<td>0.0098</td>
<td>0.0102</td>
<td>0.0115</td>
</tr>
<tr>
<td>LIN</td>
<td>0.0077</td>
<td>0.0075</td>
<td>0.01</td>
<td>0.0075</td>
<td>0.0077</td>
<td>0.0093</td>
</tr>
<tr>
<td>MRK</td>
<td>0.0071</td>
<td>0.0071</td>
<td>0.0112</td>
<td>0.0071</td>
<td>0.0073</td>
<td>0.0087</td>
</tr>
<tr>
<td>SAP</td>
<td>0.007</td>
<td>0.0078</td>
<td>0.013</td>
<td>0.0078</td>
<td>0.0072</td>
<td>0.0086</td>
</tr>
<tr>
<td>ADM</td>
<td>0.0084</td>
<td>0.0083</td>
<td>0.0154</td>
<td>0.0083</td>
<td>0.0087</td>
<td>0.0105</td>
</tr>
<tr>
<td>AHT</td>
<td>0.0153</td>
<td>0.0153</td>
<td>0.0215</td>
<td>0.0152</td>
<td>0.0162</td>
<td>0.0196</td>
</tr>
<tr>
<td>BA</td>
<td>0.0089</td>
<td>0.0087</td>
<td>0.0134</td>
<td>0.0088</td>
<td>0.0093</td>
<td>0.0106</td>
</tr>
<tr>
<td>BP</td>
<td>0.0076</td>
<td>0.0077</td>
<td>0.0194</td>
<td>0.0077</td>
<td>0.0076</td>
<td>0.0089</td>
</tr>
<tr>
<td>CCL</td>
<td>0.0081</td>
<td>0.0082</td>
<td>0.0156</td>
<td>0.0082</td>
<td>0.0085</td>
<td>0.0106</td>
</tr>
<tr>
<td>IAG</td>
<td>0.021</td>
<td>0.0213</td>
<td>0.0322</td>
<td>0.0212</td>
<td>0.0214</td>
<td>0.0246</td>
</tr>
<tr>
<td>SKY</td>
<td>0.0323</td>
<td>0.0323</td>
<td>0.0339</td>
<td>0.0322</td>
<td>0.0325</td>
<td>0.0398</td>
</tr>
<tr>
<td>VOD</td>
<td>0.0072</td>
<td>0.0083</td>
<td>0.0166</td>
<td>0.0081</td>
<td>0.0074</td>
<td>0.0088</td>
</tr>
<tr>
<td>AAL</td>
<td>0.0139</td>
<td>0.0149</td>
<td>0.0228</td>
<td>0.0148</td>
<td>0.0155</td>
<td>0.0181</td>
</tr>
<tr>
<td>AAPL</td>
<td>0.0086</td>
<td>0.0086</td>
<td>0.0224</td>
<td>0.0085</td>
<td>0.0088</td>
<td>0.0095</td>
</tr>
<tr>
<td>AMZN</td>
<td>0.0095</td>
<td>0.0102</td>
<td>0.0232</td>
<td>0.01</td>
<td>0.0101</td>
<td>0.0115</td>
</tr>
<tr>
<td>C</td>
<td>0.0087</td>
<td>0.0094</td>
<td>0.0417</td>
<td>0.0094</td>
<td>0.0097</td>
<td>0.0123</td>
</tr>
<tr>
<td>GOOGL</td>
<td>0.008</td>
<td>0.0082</td>
<td>0.0098</td>
<td>0.0081</td>
<td>0.0082</td>
<td>0.0091</td>
</tr>
<tr>
<td>MSFT</td>
<td>0.0077</td>
<td>0.0074</td>
<td>0.01</td>
<td>0.0075</td>
<td>0.0078</td>
<td>0.009</td>
</tr>
<tr>
<td>SPY</td>
<td>0.0039</td>
<td>0.0038</td>
<td>0.0057</td>
<td>0.0038</td>
<td>0.0041</td>
<td>0.005</td>
</tr>
<tr>
<td>T</td>
<td>0.0083</td>
<td>0.0081</td>
<td>0.0105</td>
<td>0.0082</td>
<td>0.0081</td>
<td>0.0086</td>
</tr>
<tr>
<td>Mean</td>
<td>0.012</td>
<td>0.013</td>
<td>0.012</td>
<td>0.013</td>
<td>0.013</td>
<td>0.015</td>
</tr>
<tr>
<td>Standard Deviation</td>
<td>0.012</td>
<td>0.012</td>
<td>0.013</td>
<td>0.012</td>
<td>0.011</td>
<td>0.014</td>
</tr>
</tbody>
</table>

The previous metrics attempt to measure the predictive power of models. However, they do not provide any insight into the performance of the compared methods from a trading point of view. Table 5 shows the SR values of each method in the considered stocks, where a higher value means greater returns for the portfolio relative to the inherent risk (Kaplanski et al., 2016). We see that our proposal outperforms the other algorithms, converging to higher values of SR, which suggests that the proposed interdependence strategy helps maximize returns while reducing volatility. This strategy selects a model based on each new data point that arrives. For example, it may be possible to obtain some predictions on ADS (a stock traded in Germany) using the model learned from VOD (a stock traded in the UK). However, the converging speed may be adversely affected if the kernel bandwidth parameter is inappropriately chosen. Consequently, a suitable value can be selected using a method like Silverman’s...
Table 4: Testing MSE at final iteration in stock returns prediction.

<table>
<thead>
<tr>
<th>Stock</th>
<th>Method</th>
<th>LSTM</th>
<th>NICE</th>
<th>Proposal</th>
<th>QKLMS</th>
<th>VAR</th>
<th>VECM</th>
</tr>
</thead>
<tbody>
<tr>
<td>DE</td>
<td>ADS</td>
<td>0.0003</td>
<td>0.0003</td>
<td>0.0004</td>
<td>0.0003</td>
<td>0.0003</td>
<td>0.0004</td>
</tr>
<tr>
<td>DE</td>
<td>ALV</td>
<td>0.0003</td>
<td>0.0003</td>
<td>0.0003</td>
<td>0.0003</td>
<td>0.0003</td>
<td>0.0003</td>
</tr>
<tr>
<td>DE</td>
<td>DPW</td>
<td>0.0135</td>
<td>0.0135</td>
<td>0.014</td>
<td>0.0134</td>
<td>0.0132</td>
<td>0.0172</td>
</tr>
<tr>
<td>DE</td>
<td>DTE</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0003</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
</tr>
<tr>
<td>DE</td>
<td>HRI</td>
<td>0.0002</td>
<td>0.0002</td>
<td>0.0004</td>
<td>0.0002</td>
<td>0.0002</td>
<td>0.0002</td>
</tr>
<tr>
<td>DE</td>
<td>LIN</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0002</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
</tr>
<tr>
<td>DE</td>
<td>MRK</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0002</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0002</td>
</tr>
<tr>
<td>DE</td>
<td>SAP</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0003</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
</tr>
<tr>
<td>UK</td>
<td>ADM</td>
<td>0.0002</td>
<td>0.0002</td>
<td>0.0004</td>
<td>0.0002</td>
<td>0.0002</td>
<td>0.0002</td>
</tr>
<tr>
<td>UK</td>
<td>AHT</td>
<td>0.0005</td>
<td>0.0005</td>
<td>0.0008</td>
<td>0.0005</td>
<td>0.0005</td>
<td>0.0007</td>
</tr>
<tr>
<td>UK</td>
<td>BA</td>
<td>0.0002</td>
<td>0.0002</td>
<td>0.0003</td>
<td>0.0002</td>
<td>0.0002</td>
<td>0.0002</td>
</tr>
<tr>
<td>UK</td>
<td>BP</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0005</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
</tr>
<tr>
<td>UK</td>
<td>CCL</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0004</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0002</td>
</tr>
<tr>
<td>UK</td>
<td>IAG</td>
<td>0.0009</td>
<td>0.0009</td>
<td>0.0016</td>
<td>0.0009</td>
<td>0.0009</td>
<td>0.0011</td>
</tr>
<tr>
<td>UK</td>
<td>SKY</td>
<td>0.0028</td>
<td>0.0028</td>
<td>0.0034</td>
<td>0.0028</td>
<td>0.0028</td>
<td>0.0037</td>
</tr>
<tr>
<td>UK</td>
<td>VOD</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0004</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0002</td>
</tr>
<tr>
<td>US</td>
<td>AAL</td>
<td>0.0004</td>
<td>0.0004</td>
<td>0.0008</td>
<td>0.0004</td>
<td>0.0004</td>
<td>0.0005</td>
</tr>
<tr>
<td>US</td>
<td>AAPL</td>
<td>0.0001</td>
<td>0.0002</td>
<td>0.0006</td>
<td>0.0002</td>
<td>0.0002</td>
<td>0.0002</td>
</tr>
<tr>
<td>US</td>
<td>AMZN</td>
<td>0.0002</td>
<td>0.0002</td>
<td>0.0007</td>
<td>0.0002</td>
<td>0.0002</td>
<td>0.0003</td>
</tr>
<tr>
<td>US</td>
<td>C</td>
<td>0.0001</td>
<td>0.0002</td>
<td>0.0019</td>
<td>0.0002</td>
<td>0.0002</td>
<td>0.0002</td>
</tr>
<tr>
<td>US</td>
<td>GOOGL</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0002</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0002</td>
</tr>
<tr>
<td>US</td>
<td>MSFT</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0002</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0002</td>
</tr>
<tr>
<td>US</td>
<td>SPY</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0001</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0001</td>
</tr>
<tr>
<td>US</td>
<td>T</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0002</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0002</td>
</tr>
<tr>
<td></td>
<td>Mean</td>
<td>0.001</td>
<td>0.001</td>
<td>0.001</td>
<td>0.001</td>
<td>0.001</td>
<td>0.001</td>
</tr>
<tr>
<td></td>
<td>Standard Deviation</td>
<td>0.003</td>
<td>0.003</td>
<td>0.003</td>
<td>0.003</td>
<td>0.003</td>
<td>0.003</td>
</tr>
</tbody>
</table>

In addition, the proposed method uses linear adaptive structures in reproducing kernel Hilbert spaces (RKHS) to obtain nonlinear filters in the input space, enhancing prediction tasks while preserving the simplicity of linear adaptive filters.

The true and predicted stock returns are shown in Figure 3 for three representative stocks, namely Allianz SE (ALV), Ashtead Group PLC (AHT), and Alphabet In-CL A (GOOGL). The left column graphs (Figures 3(a), 3(c) and 3(e)) display predictions for all tested methods. The right column graphs (Figures 3(b), 3(d) and 3(f)) show the predictions of our proposal, where each color represents a different stock market. Note that, unlike the other methods, the proposed framework obtains more accurate predictions when predicting abrupt stock return changes. Figure 3(b) provides an explanation for this behavior, where it can be seen that predictions were calculated using models learned from stocks in the US and UK. Thus, the interdependence strategy of
Table 5: Testing SR at final iteration in stock returns prediction.

<table>
<thead>
<tr>
<th>Stock</th>
<th>Method</th>
<th>LSTM</th>
<th>NICE</th>
<th>Proposal</th>
<th>QKLMS</th>
<th>VAR</th>
<th>VECM</th>
</tr>
</thead>
<tbody>
<tr>
<td>DE</td>
<td>ADS</td>
<td>0.9017</td>
<td>0.2162</td>
<td>3.6583</td>
<td>0.0051</td>
<td>2.2682</td>
<td>0.9369</td>
</tr>
<tr>
<td></td>
<td>ALV</td>
<td>1.2965</td>
<td>0.2089</td>
<td>3.3982</td>
<td>0.0729</td>
<td>1.0817</td>
<td>0.0621</td>
</tr>
<tr>
<td></td>
<td>DPW</td>
<td>-0.7522</td>
<td>-1.0027</td>
<td>2.13</td>
<td>-1.1187</td>
<td>-0.8717</td>
<td>0.7948</td>
</tr>
<tr>
<td></td>
<td>DTE</td>
<td>0.2688</td>
<td>-0.0372</td>
<td>1.9498</td>
<td>-0.1236</td>
<td>0.7665</td>
<td>1.2571</td>
</tr>
<tr>
<td></td>
<td>HEI</td>
<td>0.8574</td>
<td>0.2089</td>
<td>2.0087</td>
<td>-0.0287</td>
<td>1.3908</td>
<td>2.0077</td>
</tr>
<tr>
<td></td>
<td>LIN</td>
<td>0.5044</td>
<td>0.1126</td>
<td>3.1285</td>
<td>0.0846</td>
<td>1.2633</td>
<td>2.5074</td>
</tr>
<tr>
<td></td>
<td>MRK</td>
<td>0.7296</td>
<td>0.0683</td>
<td>3.627</td>
<td>-0.0257</td>
<td>1.4655</td>
<td>0.6756</td>
</tr>
<tr>
<td></td>
<td>SAP</td>
<td>0.4705</td>
<td>0.0497</td>
<td>4.39</td>
<td>-0.0371</td>
<td>1.8604</td>
<td>2.4806</td>
</tr>
<tr>
<td>UK</td>
<td>ADM</td>
<td>0.4663</td>
<td>0.334</td>
<td>5.9617</td>
<td>0.0065</td>
<td>2.4326</td>
<td>1.0164</td>
</tr>
<tr>
<td></td>
<td>AHT</td>
<td>2.2794</td>
<td>0.936</td>
<td>5.3711</td>
<td>-1.2048</td>
<td>-1.3246</td>
<td>-0.3778</td>
</tr>
<tr>
<td></td>
<td>BA</td>
<td>0.7246</td>
<td>0.0998</td>
<td>2.1991</td>
<td>-0.0579</td>
<td>1.2907</td>
<td>1.2656</td>
</tr>
<tr>
<td></td>
<td>BP</td>
<td>1.0361</td>
<td>1.5645</td>
<td>6.549</td>
<td>-0.1094</td>
<td>-0.0275</td>
<td>-1.0334</td>
</tr>
<tr>
<td></td>
<td>CCL</td>
<td>1.2433</td>
<td>0.5456</td>
<td>5.9871</td>
<td>0.0216</td>
<td>-0.7435</td>
<td>-0.5154</td>
</tr>
<tr>
<td></td>
<td>IAG</td>
<td>1.9193</td>
<td>-1.123</td>
<td>4.5652</td>
<td>1.5662</td>
<td>0.3662</td>
<td>-1.3497</td>
</tr>
<tr>
<td></td>
<td>SKY</td>
<td>-0.2469</td>
<td>0.3637</td>
<td>1.4274</td>
<td>0.318</td>
<td>-1.2898</td>
<td>-1.3334</td>
</tr>
<tr>
<td></td>
<td>VOD</td>
<td>0.6955</td>
<td>0.222</td>
<td>5.2958</td>
<td>0.1688</td>
<td>0.3891</td>
<td>1.4227</td>
</tr>
<tr>
<td>US</td>
<td>AAL</td>
<td>1.1319</td>
<td>-0.5774</td>
<td>5.8926</td>
<td>0.4946</td>
<td>-1.144</td>
<td>-1.1616</td>
</tr>
<tr>
<td></td>
<td>AAPL</td>
<td>0.6432</td>
<td>0.1279</td>
<td>2.6692</td>
<td>-0.1274</td>
<td>1.0422</td>
<td>1.648</td>
</tr>
<tr>
<td></td>
<td>AMZN</td>
<td>0.5934</td>
<td>0.1681</td>
<td>1.5527</td>
<td>-0.0205</td>
<td>1.0327</td>
<td>1.7307</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>0.975</td>
<td>0.2737</td>
<td>3.6621</td>
<td>0.0879</td>
<td>1.1656</td>
<td>-0.3031</td>
</tr>
<tr>
<td></td>
<td>GOOGL</td>
<td>0.5118</td>
<td>0.1185</td>
<td>2.9763</td>
<td>0.0459</td>
<td>1.3334</td>
<td>1.2835</td>
</tr>
<tr>
<td></td>
<td>MSFT</td>
<td>0.5291</td>
<td>0.1301</td>
<td>2.8933</td>
<td>0.0355</td>
<td>1.2268</td>
<td>2.1107</td>
</tr>
<tr>
<td></td>
<td>SPY</td>
<td>0.4121</td>
<td>0.0827</td>
<td>1.7878</td>
<td>-0.0977</td>
<td>1.0711</td>
<td>1.7259</td>
</tr>
<tr>
<td></td>
<td>T</td>
<td>1.2947</td>
<td>0.1438</td>
<td>3.3456</td>
<td>-0.0684</td>
<td>1.3681</td>
<td>1.1828</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mean</th>
<th>0.77</th>
<th>0.136</th>
<th>3.6*</th>
<th>0.004</th>
<th>0.722</th>
<th>0.738</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard Deviation</td>
<td>0.606</td>
<td>0.517</td>
<td>1.532</td>
<td>0.484</td>
<td>1.061</td>
<td>1.207</td>
</tr>
</tbody>
</table>

our proposal provides an advantage over competition algorithms, as they do not consider the interconnections between stock markets. Figure 3(d) shows a similar situation, where predictions are improved using models learned from the US. Figure 3(f) further suggests that the predictions of a particular stock can be enhanced by incorporating models learned from other stock markets.

Figure 4 shows the models used by the proposed framework to predict the considered stocks, where each color represents a different stock market. Figure 4(a) displays the percentage of models from Germany, the UK and the US that were used to predict stock returns. For example, 18% of the models used to predict the 8 stocks traded in Germany are from the UK, while 60% of the models used to predict the 8 stocks traded in the US are from Germany. The models from Germany, such as SAP AG (SAP) (see Figure 4(b)), are widely used to predict stock returns in the UK and the US (see Figure 4(a)), suggesting that the patterns encoded by German stocks are more appropriate to predict
Figure 3: Stock return predictions in the test sets of three representative stocks.

other stock markets. Figure 4(b) also shows that there are some stocks with 0% such as ADS, DTE, LIN, SKY, AAL, etc. This means that their models were
not used to obtain their own predictions, nor were they used to predict other stocks. However, their performance is still competitive (see Tables 3 to 5).

![Graph (a) Markets](image)

![Graph (b) Stocks](image)

Figure 4: Percentage of models used by the proposed framework to predict the 24 stocks.

**Note:** The Python implementation can be downloaded from [https://segarciave.github.io/stock_returns_prediction](https://segarciave.github.io/stock_returns_prediction).

6. Conclusions and Future Work

This study introduces a framework for stock returns prediction using KAF within a stock market interdependence approach. The framework sequentially predicts stock returns by considering interconnections between stock markets. The proposed sequential learning strategy uses the data distribution as the criterion to encode different patterns of the input space, which in some circumstances may yield a more compact network. This strategy helps maximize...
returns and reduces volatility while maintaining the robustness and simplicity of kernel-based adaptive filters. The proposed framework for stock returns prediction comprises an $\mathcal{O}(N^3)$ computational complexity, which results in an expensive execution time when compared to conventional algorithms. Thus, more elaborate optimization algorithms or GPU-based implementations must be considered to reduce the computational burden.

The framework has been tested on 24 different stocks from three major economies. Simulation results demonstrate that the interdependence strategy used by our proposal enhances prediction accuracy, representing an advantage over compared methods. In addition, as the simulation results show, the models learned from the German market are more suitable for making predictions in other stock markets. This suggests that the US market is more influenced by the European and not vice versa, which is in line with previous empirical findings (Jizba et al., 2012; Rezayat & Yavas, 2006).

We are in the process of expanding our research in the following areas: i) considering other stock markets and portfolios of financial assets such as commodities and bonds; ii) considering additional Mercer kernels, i.e., not restricted to the Gaussian kernel; iii) introducing a hyper-parameter tuning procedure into the proposed method.

Acknowledgment

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 675044.

References


Chai, T., & Draxler, R. R. (2014). Root mean square error (rmse) or mean absolute error (mae)?-arguments against avoiding rmse in the literature. *Geoscientific model development, 7*, 1247–1250.


