International Journal of Data Science and Analytics
https://doi.org/10.1007/s41060-025-00854-4

REGULAR PAPER

Machine learning, stock market forecasting, and market efficiency: a comparative study

Oscar Bustos[1] · Alexandra Pomares-Quimbaya[1] · Rémi Stellian[2]

Received: 7 October 2024 / Accepted: 16 June 2025

Abstract
Analyses of the accuracy of machine learning algorithms for predicting stock market indices typically employ only one or a small set of stock indices and a small number of time periods. Moreover, these analyses generally ignore the influence of market efficiency on algorithm accuracy, despite theoretical evidence of the potential for such an effect. This paper aims to fill this gap by proposing and applying a method focused on the analysis of the accuracy of stock market prediction models in comparison with market efficiency. This method is applied to a dataset comprising 55 markets and 65 periods to compare the most accurate and frequently used algorithms for stock index prediction: support vector machines, artificial neural networks, gradient-boosted trees, Naive Bayes, random forest, logistic regression, and long-short term memory. These algorithms were trained to predict stock market indices using technical indicators as input. After analyzing the algorithms' results, we present a detailed comparative analysis of their accuracy supported by a set of nonparametric measures. In addition, dynamic panel data analysis is used to determine whether there is a relationship between stock market's efficiency and algorithmic precision. We highlight the algorithms that tend to show the best performance in terms of accuracy and obtain interesting results regarding market efficiency.

Keywords Stock market forecast · Machine learning · Financial modeling · Market efficiency · Dynamic panel data analysis

1 Introduction

The ability to predict stock market movements has important implications for investors, researchers and pension funds, since it helps create strategies that reduce risks and maximize returns [1]. However, accurately forecasting stock market movements is a challenging task, making it an active area of research in economics, computer science and finance [2–4]. Machine learning algorithms are now the primary tool for stock market prediction, with traditional statistical methods playing only a minor role [2]. The main input data are technical indicators calculated based on historical prices, whereas text mining and fundamental analysis based on firm attributes such as size, capitalization, price-earnings ratio, assets, and liabilities are of lesser interest. Compared with traditional statistical methods, machine learning algorithms (hereafter referred to as "algorithms" for the sake of simplicity) are better able to capture the complex nature of financial data.

There is no consensus on the most accurate algorithms for stock market prediction [1]. Sophisticated algorithms, such as ensemble models—which combine various algorithms instead of using a single algorithm, as exemplified by gradient-boosted trees—and deep learning models—a subset of artificial neural networks—are widely employed in the literature [5]. However, some studies suggest that less complex algorithms, such as support vector machines, have better performance than gradient-boosted trees and artificial neural networks [6, 7].

There are two potential explanations for the lack of consensus on machine learning for stock market prediction. First, many evaluations of these algorithms use a small sample of stock markets or even a single market. A comprehensive systematic review by [3] revealed that studies of stock mar-

✉ Oscar Bustos
oscar.bustos@javeriana.edu.co

Alexandra Pomares-Quimbaya
pomares@javeriana.edu.co

Rémi Stellian
rstellian@javeriana.edu.co

1 Department of Systems Engineering, Pontificia Universidad Javeriana, Bogotá, Colombia

2 Department of Business Administration, Pontificia Universidad Javeriana, Bogotá, Colombia

Published online: 25 July 2025

ket forecasting concentrate on just three countries: the USA (62 studies of the Dow Jones Industrial Average (DJIA), S&P500, and NASDAQ), Taiwan (28 studies of the Taiwan Capitalization Weighted Stock Index (TAIEX) and Taiwan Stock Exchange Index Futures (FITX)), and China (25 studies of the Shanghai Stock Exchange (SSE) and China Stock Index (CSI)). Second, most studies of stock market prediction test algorithms over a small time span. According to [5], the average time span is 5 years, and even half of studies use a time span of 3 years or less. To the best of our knowledge, no work has compared the accuracy of algorithms on a large panel of stock markets over a sufficiently long time span to obtain more robust results from an empirical standpoint. Therefore, the literature would benefit from a study that addresses this research gap and ultimately provides findings about machine learning and stock markets that are more general across markets and across time.

In addition, theory suggests that *market efficiency* has a decisive impact on the ability to forecast price movements. Indeed, if market efficiency implies that price forecasting is more difficult [8], then we should expect that markets that are more efficient deteriorate the ability to predict prices by means of machine learning. However, to the best of our knowledge, available studies about machine learning and stock market prediction have largely omitted the influence of market efficiency. An exception is [9], who investigate whether the nearest-neighbors algorithm makes more accurate predictions in efficient markets than in non-efficient markets. Nonetheless, the study by [9] is no longer representative of current trends in machine learning, because since then new algorithms have been created and are widely employed in the literature. Ultimately, there is a lack of empirical evidence about the influence of market efficiency on the accuracy of machine learning for predicting stock market prices. It is worth going back to this issue with an analysis that, in addition, rests upon a large panel of stock markets over a sufficiently long time span, as explained before.

This paper simultaneously addresses these three shortcomings of the literature, i.e., small sample of markets, short time spans, and neglect of market efficiency, by employing an analytical approach that systematically compares the performance of various algorithms in predicting price movements of diverse stock market indices. Our approach includes an analysis of the impact of market efficiency on accuracy. In addition to the standard measurement of market efficiency given by the Hurst exponent [10], we employ the Hall–Wood (HW) metric [11] and the Robust Genton (RG) estimator [12]. We study the following algorithms: support vector machines (SVM), artificial neural networks (ANN), random forest (RF), gradient-boosted trees (GB), Naive Bayes (NB), logistic regression (LR), and long-short term memory (LSTM). Ultimately, we apply these algorithms to predict the prices of stock market indices based on technical indicators for a panel of 55 stock markets over a time span of 65 two-month periods from 2011/01 to 2022/07.

The rest of the paper is organized as follows. Section 2 introduces the concept of market efficiency, underscores its significance for stock market prediction, and explores commonly employed estimators of market efficiency. Section 3 reviews relevant works that use machine learning to predict the price movements of stock market indices. Section 4 describes the methodology, including the type of information used to train the algorithms, the corresponding data preprocessing, and the parametrization of the algorithms. Section 5 compares the relative accuracies of the algorithms in relation to market efficiency. Finally, Sect. 6 provides concluding remarks.

2 Background: market efficiency

According to [8], abnormal risk-adjusted returns are not possible if markets are efficient. Markets are defined as efficient if prices incorporate all available information completely, instantly, and without cost. Assuming that the flow of new information is a random variable, movements in prices are also random [13]. Such randomness hinders market prediction and is the reason why market efficiency is incorporated in the theorization of prices as stochastic variables, for example random walks or martingales [14]. Because the efficient market hypothesis is somewhat extreme, different categories of efficiency have been suggested [15, 16]. Weak efficiency is defined as the uselessness of past prices for outperforming markets. Semi-strong efficiency includes weak efficiency and makes publicly available information useless for outperforming markets. Last, strong efficiency includes semi-strong efficiency and prevents private “insider” information from offering arbitrage opportunities that would give rise to abnormal risk-adjusted returns. In general, weaker efficiency reduces the dependence of prices on new inflows of information and therefore the randomness of prices. Ultimately, from a theoretical point of view, weaker efficiency may improve the ability of investors to predict price movements. This gives investors an information advantage, making it easier to gain higher-than-average returns. The converse is true when efficiency is stronger.

Therefore, theory suggests that algorithms make less accurate predictions regarding stock prices if markets are more efficient. Because empirical evidence about this proposition is scarce, as already stated by the introduction, it is about time to test if machine learning loses some of its ability to predict stock prices if the corresponding stock markets are more efficient. To this purpose, it is necessary to quantify market efficiency. The rest of the section describes the standard quantification of market efficiency—the Hurst exponent—and two additional metrics in the form of fractal dimensions, namely the Hall–Wood estimator and the highly robust variogram.

2.1 Hurst exponent estimator

Several financial articles [9, 10, 17] quantify market efficiency using the Hurst exponent, which was first proposed by [18] in the field of hydrology to model the long-term auto-correlation of water levels in the Nile River. One of the most popular methods of calculating the Hurst exponent H is re-scaled range analysis (R/S analysis) [19]. This method splits the time series into d subseries of length n and calculates the standardized cumulative time series $(R/S)_n$, which asymptotically follows the relation $(R/S)_n \approx c \cdot n^H$;

$$\begin{cases} (R/S)_n = \frac{1}{d}\sum_{m=1}^{d} \dfrac{R_m}{S_m} \text{ with:} \\ R_m = \max(Y_{1,m}, \ldots, Y_{n,m}) - \min(Y_{1,m}, \ldots, Y_{n,m}); \\ \quad \text{and } S_m = sd(X_m) \end{cases} \tag{1}$$

where $X_{i,m}$ is the i-th point in subseries m ($i = 1, \ldots, n$ and $m = 1, \ldots, d$); $Y_{i,m} = X_{i,m} - \bar{X}_m$ is the normalized value of $X_{i,m}$ by subtracting the expected value of $\{X_{1,m}, X_{2,m}, \ldots, X_{n,m}\}$; $Y_{i,m}$ is the cumulative time series $\sum_{j=1}^{i} Y_{j,m}$. The Hurst exponent is calculated by linear regression of the n data points in Eq. (2). Thereafter:

$$\log(R/S)_n = \log(c) + H \cdot \log(n) \tag{2}$$

We use the R function *hurstexp* from the Pracma library [20] to estimate the Hurst exponent.

2.2 Fractal estimators

Two fractal dimensions estimators are employed in this study to quantify market efficiency. The first fractal dimension estimator is the Hall–Wood estimator described by [11]. This metric is based on the box-counting estimators of fractal dimension technique, which divides a time series X with n regularly spaced points into q boxes B_{ij} and calculates the areas of the boxes according to Eq. (3):

$$\begin{cases} A(l) = \sum A_{il} = \varepsilon_l \sum (U_{il} - V_{il}) \text{ with:} \\ U_{il} = \max_{j \in B_{il}} X\left(\frac{j}{n}\right); V_{il} = \min_{j \in B_{il}} X\left(\frac{j}{n}\right); \\ \quad \text{and } \varepsilon_l = \dfrac{lm}{n} \end{cases} \tag{3}$$

The fractal dimension $\hat{D}_{HW}$ is estimated using ordinary least squares over $\log A(l)$ as stated in Eq. (4):

$$\hat{D}_{HW} = 2 - \sum_{l=1}^{k} \frac{(X_l - \bar{X}) \log A(l)}{(X_l - \bar{X})^2} \tag{4}$$

To calculate the HW metric, we use the R function *fd.estim.hallwood* from the Fractaldim library [21] with $l \leq 2$ as the default parameter.

The second fractal dimension estimator that quantify market efficiency is the RG metric described by [12], which is also known as the highly robust variogram. This metric is based on the method of moments for estimating a variogram robust to outliers. The fractal dimension D is approximated using the estimator $\widehat{D}_{RG}$ calculated with the following equation, where $\widehat{V}_2(l/n)$ is the estimator of the variogram for n segments in a time series X:

$$\begin{cases} \widehat{D}_{RG} = 2 - \dfrac{\log(\widehat{V}_2(2/n)) - \log(\widehat{V}_2(l/n))}{2\log(2)} \\ \text{with } \widehat{V}_2(l/n) = \frac{1}{2(l-n)} \sum_{i=1}^{n} \left(X_{i/n} - X_{(i-l)/n}\right)^2 \end{cases} \tag{5}$$

3 Related works

Several articles use machine learning for stock market prediction. In a summary of articles published between 2000 and 2019 that investigate the application of machine learning to stock market prediction, [3] highlight articles reporting the highest precision for each algorithm. Our article expands this list with a selection of works from 2020 to 2023. As shown in Table 1, the accuracy rates of algorithms change depending on the stock market. Whereas [22] and [23] report accuracy rates above 90%, [24, 25] observe accuracy rates close to 50%. This large variability in accuracy rates in the literature is perplexing.

Nelson et al. [32] use deep learning algorithms to generate predictive models for the IBovespa index (Brazil). In particular, they compare LSTM, which takes advantage of information from the technical indicators of the IBovespa index, with multilayer perceptron (MLP) and RF. They conclude that LSTM has considerably greater accuracy. Dingli et al. [29] model three major stock indices in the USA: the S&P500, DJIA, and NASDAQ-100. Using technical indicators as input, they generate a model with 60% accuracy for the next week's price direction. Dingli et al. [29] also conclude that LSTM outperforms other algorithms, such as SVM and LR. Minh et al. [31] use two-stream gated recurrent unit (GRU) and a model called Stock2Vec to process financial news, forecasting the S&P 500 index with an accuracy of 66.32%. Jing et al. [30] propose a deep learning approach using LSTM with a sentiment analysis model for stock price prediction for six key industries on the Shanghai Stock Exchange (SSE), achieving an F-measure of 84.82%. Kim et al. [24] forecast the direction of US stock prices by including a technique called time-varying Effective Transfer Entropy (ETE). They compare LSTM with baseline algorithms such as LR, MLP, RF, and XGBoost, and find that deep

Table 1 Summary of Related Works

Author	Algorithm	Market	Measure	Result
[26]	ANN	S&P500, DJIA	Hit ratio	86.81%, 88.98%
[27]	ANN	Nikkei 225	Hit ratio	81.27%
[28]	ANN	NASDAQ, NYSE, and S&P500	Accuracy	56% on average
[29]	Deep learning	S&P500, DJIA, NASDAQ-100	Accuracy	60% on average
[30]	Deep learning	SSE	F-Measure	84.82%
[24]	Deep learning	S&P500	Accuracy	53.53%
[25]	Deep learning	CSI 300	Accuracy	51.43%
[31]	Deep learning	S&P500	Accuracy	66.32%
[32]	Deep learning	IBovespa	Accuracy	55.9%
[22]	Deep learning	Chinese Stock Market	Accuracy	93.25% on average
[33]	Ensemble	S&P500	Accuracy	60%
[34]	Ensemble	Chinese A-shares	Accuracy	52.93%
[23]	Ensemble	KOSPI	Accuracy	93.28%
[35]	Ensemble	S&P500, Dow30, NASDAQ	Accuracy	69.17%, 68.86%, 70.74%
[9]	Other	60 stock indices	Accuracy	Depending on index
[36]	Other	TSE return	Accuracy	80.24%
[37]	Other	AAPL Stock	Accuracy	85.8%
[38]	Other	Enron Stock	Accuracy	86.67%
[39]	Other	S&P, BSE, SENSEX, Nifty 50	Accuracy	80% on average
Proposed approach	7 different models	55 Stock Indices	Accuracy	Sect. 5.1

learning algorithm achieves the highest accuracy (53.53%). In summary, the accuracy of deep learning varies greatly depending on the stock market.

Nonetheless, despite the growing popularity of deep learning, [3] find that several studies report superior performance by ensemble algorithms, such as random forest (RF) and gradient-boosted trees (GB). Carta et al. [33] model the S&P500 using news information, which is processed through a custom lexicon for the stock market. Several ensemble algorithms were compared, such as RF and GB. Carta et al. [33] found that a decision tree algorithm achieves the highest performance, with an accuracy close to 60%. Yuan et al. [34] predicts the Chinese A-share market using a set of technical and macroeconomic indicators as input. They compare the accuracy of the RF with algorithms such as SVM or ANN. RF gives the best result with 52.93%. Jiang et al. [35] seeks to predict the main stock indices of the USA (S&P500, Dow30 and NASDAQ) using a stacking framework that combines ensemble algorithms, such as RF and GB, with deep learning algorithms, such as recurrent neural networks (RNNs), bidirectional RNN, RNN with long short-term memory (LSTM) and gated recurrent unit (GRU) layer. They find that the stacking-based algorithm achieves accuracies of 69.17% for the S&P500, 68.86% for the Dow30, and 70.74% for NASDAQ. Yun et al. [23] propose an optimized GB prediction system using genetic algorithms (GAs) to predict the KOSPI stock index and obtain a prediction accuracy of 93.28%.

Despite this literature evidence of the predictive power of machine learning for different stock indices, the most suitable algorithms across a large set of stock markets remain unidentified. It is not clear how algorithms that perform well in the market(s) for which they were built or tested perform in other markets. The only comparison of machine learning algorithms applied to various stock markets is presented by [9], who analyze 60 major stock indices from different countries. For each country, the ability of nearest-neighbor algorithms to predict the direction of price changes is compared. The empirical findings show that the precision of the algorithms differs between emerging markets and developed markets. Additionally, the degree of efficiency of a stock market as measured by the Hurst exponent is strongly correlated with model precision. However, the work by [9] is outdated, as applications of the most accurate techniques to the stock market, such as LSTM, were first reported in 2017 [3]. Furthermore, the time spans employed in the literature are short, which prevents the corresponding results from being sufficiently representative.

The present study is guided by the following research questions. On the basis of a large sample of markets and a large number of trading days:

- Which algorithm has the greatest prediction accuracy?
- Is there a relationship between the prediction accuracy of an algorithm and market efficiency?

Remind that, according to the market efficiency hypothesis, prices reflect all available information about a market or a specific stock at any given time. Consequently, because inflows of new information are random, prices become random in efficient markets, which makes it difficult or even impossible to predict prices. Weaker efficiency in financial markets is supposed to improve the ability to make accurate predictions regarding stock prices. To assess if algorithms improve their stock price predictions if markets are less efficient, the Hurst exponent (Eqs. 1 and 2), the Hall–Wood estimator (Eqs. 3 and 4) and the highly robust variogram (Eq. 5) constitute the main metrics to quantify market efficiency.

4 Methodology

We apply three layers of processing and modeling, as shown in Fig. 1. The first layer is the acquisition of information from a broad sample of stock indices and trading days. The second layer is the creation of two datasets: a labeled dataset suitable for training the algorithms and a dataset with the efficiency estimates of each stock index. Finally, the sample of indices is analyzed using the most commonly used algorithms in the literature on stock market forecasting. To identify the best-performing algorithms, we implement a new set of measures instead of relying solely on averages, contrary to the convention in the literature. This analysis also determines whether there is a relationship between stock market efficiency and algorithmic precision using dynamic panel data analysis.

The models were trained and tested in each period, and all efficiency estimators were calculated for each period. As each stock market index is associated with a single market, we use the terms "stock market index", "stock market", and "market" interchangeably.

4.1 Stock market data acquisition

Daily closing prices from 55 stock market indices from January 2010 to July 2022 were downloaded from the website *investing.com*. Table 2 describes the selected stock indices and the corresponding countries. Stock markets were selected not only according to their importance in terms of trading volume, but also to build a sample of stock markets with sufficient heterogeneity to reflect the diversity of stock markets around the world. In addition, over the aforementioned time span, which covers more than 3000 trading days, stock prices in each market is available and enable the calculation of the technical indicators described in Sect. 4.2. Ultimately, it is possible to build a database that describes the accuracy rate of each algorithm under consideration in each market and each period. The resulting panel data are strongly balanced, eliminating the problem of missing value imputation.[1]

4.2 Technical indicators computation

After a careful review of related works that modeled stock market, we found that most of those used technical indicators as input features [28, 29, 32, 40–42]. This work selected the most common technical indicators for effectively predicting the stock market (see Table 3). An advantage of technical indicators is that they are computed solely on the basis of historical prices, which are widely available and objective data. The following R functions from the TTR library [43] were used to calculate the technical indicators: *SMA* (simple moving average), *WMA* (weighted moving average), *RSI* (relative strength index), stochOSC (to calculate K and D indicators), WPR (*LW* or William's %R indicator), *momentum* (MOM), and *MACD* (moving average convergence divergence). Figure 2 illustrates the raw data analyzed in this article in the case of the S&P500.

The method implemented in this paper considers stock market prediction a classification problem in which the goal is to know whether the closing price of the stock will rise or fall compared with historical prices. This kind of forecast provides useful guidance about the daily long and short positions that an investor should take with respect to funds that track stock indices (exchange-traded funds and mutual funds), and hedges against potential losses using derivatives whose underlying assets are stock indices.[2] (index options and index futures). Portfolio allocation is improved as promising markets are better identified and exposure to underperforming markets can be reduced, especially for institutional investors who can afford the computational resources indispensable for machine learning [45]. Prediction can be employed for manual trading strategies or even algorithmic (or "automated") strategies [46]. The closing price, which is the target variable, is transformed into a binary variable that takes a value of zero if the closing price decreases or holds and one if the closing price increases.

[1] In practice, some prices might be missing because on the corresponding day, the market was shut down in response to an anomaly (significant increase or decrease in stock prices). This would prevent from calculating the subsequent technical indicators. The algorithms could not use these data to train or make predictions. Nevertheless, because no data were missing, anomalies do not affect our database.

[2] Our focus is on closing prices and not continuous forecasts in real time, as required by high-frequency trading. High-frequency trading benefits from AI technologies, including machine learning, yet requires a huge infrastructure and raises ethical concerns [44]

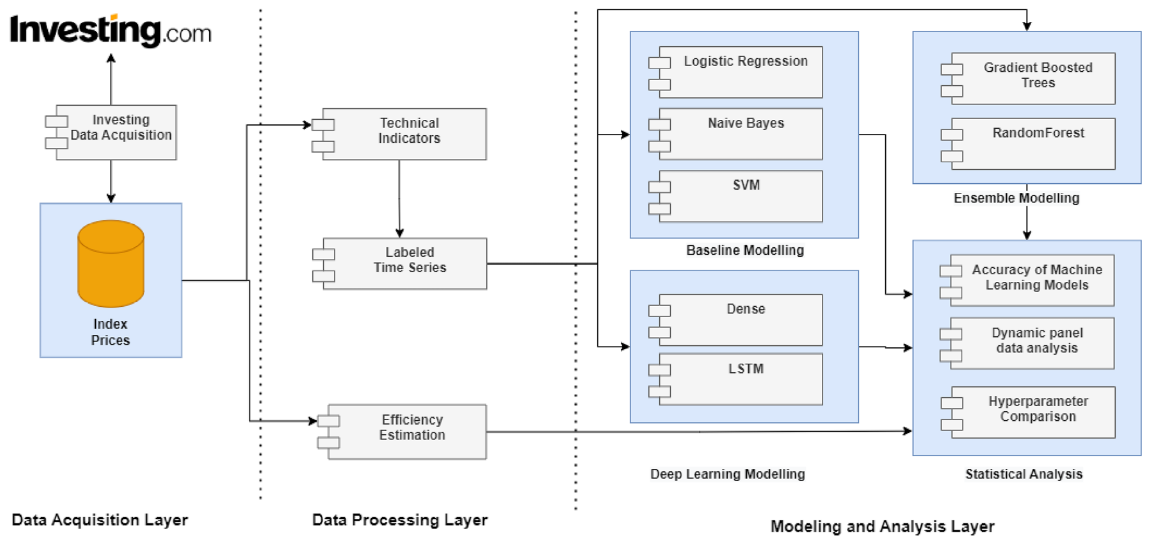

Fig. 1 Stock Market Methodology

Table 2 Stock market sample

	Stock market	Country		Stock market	Country
1	AEX	The Netherlands	29	Moroccan AS	Morocco
2	Al-Quds	Palestine	30	NZX 50	New Zealand
3	Amman SE General	Jordan	31	NZX MidCap	New Zealand
4	BEL 20	Belgium	32	NASDAQ 100	USA
5	BIST 100	Turkey	33	NASDAQ	USA
6	BSE DC	Botswana	34	Nifty 50	India
7	Blue-Chip SBITOP	Slovenia	35	Nikkei 225	Japan
8	Bovespa	Brazil	36	OMX Helsinki 25	Finland
9	CAC 40	France	37	OMXC20	Denmark
10	COLCAP	Colombia	38	OMXS30	Sweden
11	CROBEX	Croatia	39	Oslo OBX	Norway
12	CSE All-Share	Sri Lanka	40	QE General	Qatar
13	CSI 1000	China	41	RTSI	Russia
14	DAX	Germany	42	S& 500	USA
15	DFM General	UAE	43	S&P CLX IPSA	Chile
16	Dow Jones	USA	44	S&P Merval	Argentina
17	EGX 30	Egypt	45	SET	Thailand
18	FTSE 100	UK	46	SMI	Switzerland
19	FTSE ADX General	UAE	47	SZSE Component	China
20	Hang Seng	China	48	Shanghai	China
21	IBEX 35	Spain	49	ZA Top 40	South Africa
22	ICEX Main	Iceland	50	TA 35	Israel
23	IDX Composite	Indonesia	51	Tadawul AS	Saudi Arabia
24	KOSPI	South Korea	52	Tunindex	Tunisia
25	Kenya NSE 20	Kenya	53	VN 30	Viet Nam
26	MOEX	Russia	54	VNI	Viet Nam
27	MSE	Malta	55	WIG20	Polonia
28	MSM 30	Oman			

Table 3 Technical Indicators for time t

Input:
C_t is the closing price of the stock or stock index
H_n is the highest price traded in a day
L_n is the lowest price traded in a day
$\mathrm{EMA}(i)_t = \mathrm{EMA}(i)_{t-1} + \alpha\,(C_t - \mathrm{EMA}(i)_{t-1})$ is the exponential moving average
UP_t is the upward price change at day t
DOWN_t is the downward price change at day t

Indicator	Equation	Indicator	Equation
SMA_t	$\frac{1}{n}\sum_{i=0}^{n-1} C_{t-i}$	MACD_t	$\mathrm{EMA}(12)_t - \mathrm{EMA}(26)_t$
WMA_t	$\frac{\sum_{i=0}^{n-1}(n-i)C_{t-i}}{\sum_{i=0}^{n-1}(n-i)}$	LW_t	$\frac{H_n - C_t}{H_n - L_n} \times 100$
K_t	$\frac{C_t - L_{t-n}}{H_{t-n} - L_{t-n}} \times 100$	MOM_t	$C_t + C_{t-10}$
D_t	$\frac{1}{n}\sum_{i=0}^{n-1} K_t$		
RSI_t	$100 - 100\left(1 + \left(\frac{1}{n}\sum_{i=0}^{n-1}\mathrm{UP}_{t-i}\right) / \left(\frac{1}{n}\sum_{i=0}^{n-1}\mathrm{DOWN}_{t-i}\right)\right)^{-1}$		

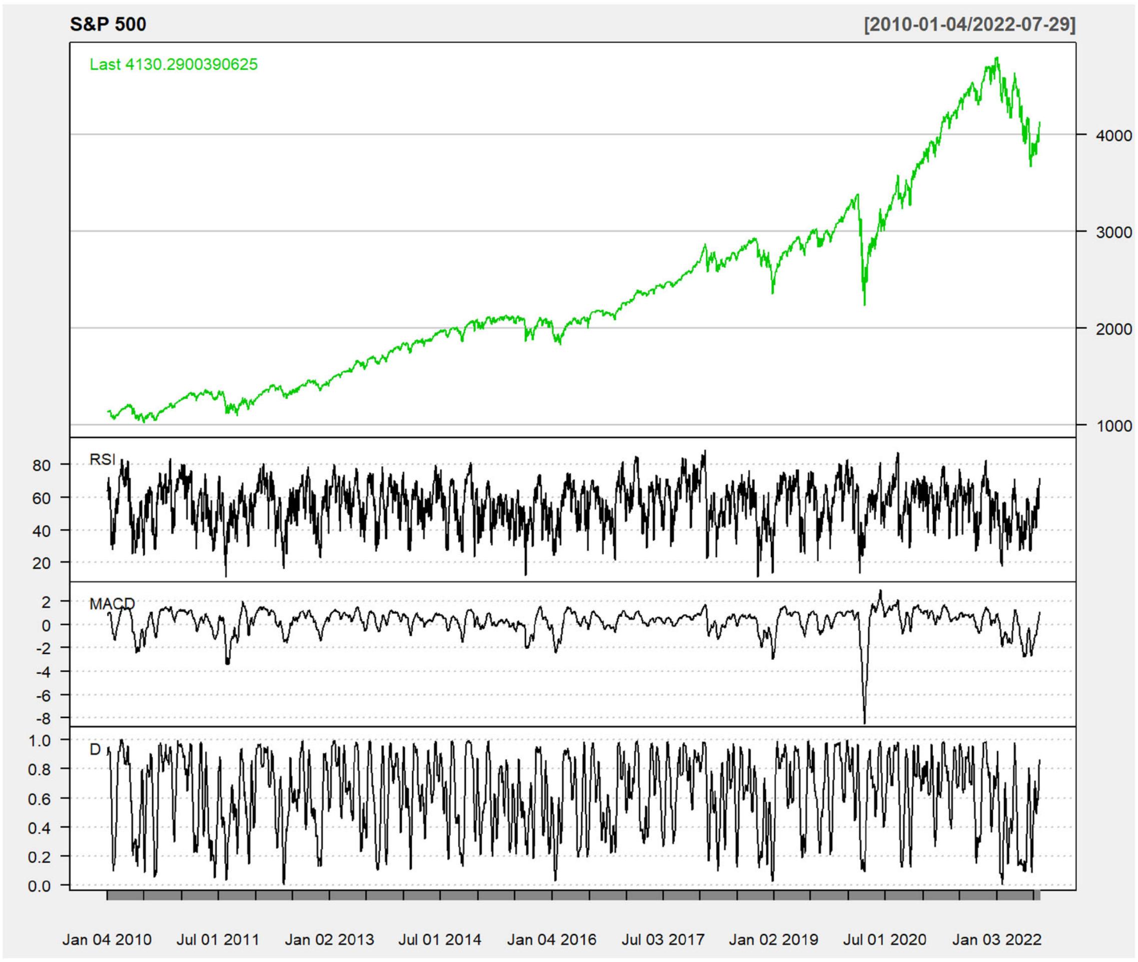

Fig. 2 S&P500 Historical Price with technical indicators

4.3 Data partitioning and labeling

The proposed method partitioned the dataset into 65 periods. Each period comprised a calendar year of training data and two months of validation data; thus, the periods overlapped in time, as shown in Fig. 3. Accuracy was calculated by feeding an algorithm with the two months of validation data that were not previously presented to it.

For each period, a set of algorithms was trained using a variation of the K-folding cross-validation technique. The data were split into K parts, and an algorithm was trained on the first k parts and tested on the $(k+1)$-*th* parts, as shown in Fig. 3. The aim is to find the best model that does not overfit the data. The train-test split inherent to the employed cross-validation technique preserves the temporal order of the data to reflect real-world forecasting scenarios. Contrary to standard cross-validation, our data partition avoids data leakage problems in which algorithms are trained using information either from a past forecast horizon or with a gap in the data series. To sum up, our study relies on data that cannot be manipulated—technical indicators—and a cross-validation technique that prevents overfitting and data leakage. This ensures an adequate implementation of machine learning where data snooping is avoided.

To train the algorithms, the Python libraries scikit-learn [47] and Tensorflow [48] were employed to preprocess the training data, select the algorithms to train, and compute the list of hyperparameters for adjustment in each of the training sessions.

Technical indicators are not valued on a standard scale. Since most algorithms are scale sensitive, Z-score transformation was used to ensure that the training data were centered on the origin and had a standard variance. The following equation was used for the transformation:

$$Z_X = \frac{X - \mu_X}{\sigma_X} \tag{6}$$

where X is the technical indicator vector used as input to train the machine learning models and μ_X and σ_X are the mean and standard deviation of the vector, respectively.

4.4 Market efficiency estimation

The use of the Hurst exponent as an estimator of the efficiency of a stock index by [9] was taken as a reference. We also calculated the HW and RG fractal dimension metrics of market efficiency presented in [49] (see Table 4 for a comparison with the Hurst exponent). The calculations were performed using the R function *fd.estim.genton* from the Fractaldim library [21].

Table 4 Efficiency Estimators

Estimator (x)	Rank	Efficiency rationale
Hurst Exponent	$[0-1]$	$x < 0.5$
Hall–Wood	$[1-2]$	$x > 1.5$
Robust Genton	$[1-2]$	$x > 1.5$

4.5 Machine learning modeling

After completing a detailed literature analysis, three categories of algorithms were selected to model the stock market: baseline algorithms, ensemble algorithms, and deep learning algorithms. In the first category, algorithms that can be trained with a low computational cost, but have performed acceptably in the literature were selected: LR, NB, and SVM. Put differently, in accordance with the literature [50], we start from the premise according to which baseline algorithms cannot be excluded right from the start for stock market prediction. The second category comprised RF and GB. Finally, the neural networks MLP and LSTM were selected for the third category.

4.5.1 Logistic regression

For binary classification, the LR model estimates the probability of the positive class using the logistic function as in Eq. (7):

$$\hat{p}(X_i) = \frac{1}{1 + \exp\left(X_i w + w_0\right)} \tag{7}$$

where X is the vector of input features, w is the vector of parameters to be estimated, and $\hat{p}$ is the estimation of the probability of belonging to the positive class. To find the parameters, the optimization problem that minimizes the following log-loss function is solved (Eq. 8):

$$\min_{w} \sum_{i=1}^{n} \left[-Y_i \log(\hat{p}(X_i)) - (1 - Y_i)\log(1 - \hat{p}(X_i))\right] \tag{8}$$

As shown in the equation, the number of parameters to be estimated is of the same order as the number of inputs. No additional hyperparameters are required, which limits the computational cost [51]. In addition, the model is easy to interpret because the parameters coincide with the importance that the model gives to each variable. Therefore, it is possible to determine which of the input variables positively or negatively affect the variable to be predicted or have no impact.

However, the LR algorithm only works well when the variable to be predicted is linearly separable using the feature

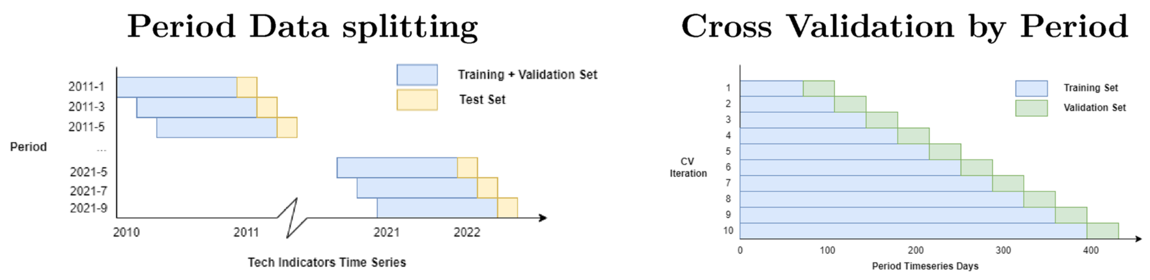

Fig. 3 Period Data splitting and Time Series Cross-Validation by Period

vector. When this separation is not possible, the algorithm has low prediction precision.

4.5.2 Naive Bayes

NB applies Bayes' conditional probability theorem to the task of classification. In this model, a very strong assumption is made about conditional independence between the predictor variables. The model is described by Eq. (9):

$$\hat{Y} = \max_{Y} P(Y) \prod_{i=1}^{n} P(X_i|Y) \tag{9}$$

where $P(Y)$ is the class probability and $P(X_i|Y)$ is the probability of the positive class given X_i. Since the model inputs are not discrete but continuous, it can be assumed that the conditional probability has a Gaussian distribution, as indicated by Eq. (10):

$$P(X_i|Y) = \frac{1}{\sqrt{2\pi\sigma_Y^2}} \exp\left(\frac{-(X_i - \mu_Y)^2}{2\sigma_Y^2}\right) \tag{10}$$

Like LR, NB is free of hyperparameters and has a low computational cost. However, in the present dataset, correlations between some of the input variables are anticipated. Consequently, the naive assumption of this model is expected to affect its accuracy.

4.5.3 Support vector machines

SVM searches for a set of hyperplanes based on the principle of maximum margin classification [51, 52]. The classification based on a hyperplane is defined by Eq. (11):

$$\begin{cases} w^T X_i - b \geq 1 \text{ when } y_i = 1 \\ w^T X_i - b \leq 1 \text{ when } y_i = 0 \end{cases} \tag{11}$$

where X_i is the i-th support vector, w is the normal vector to the hyperplane, and b is a constant. To find the value of these parameters, the optimization problem that minimizes the following hinge-loss function must be solved (Eq. 12).

$$\min_{w,b} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \left[\max\left(0, 1 - Y_i(w^T \phi(X_i) + b)\right)\right] \tag{12}$$

where C is the regularization constant and ϕ is the kernel function. The kernel function allows for generating nonlinear classification regions. In the case of a linear SVM, the kernel function is the identity function. However, the most popular kernel functions are the polynomial kernel ("poly" in Eq. 13) and the kernel based on a Gaussian Radial Basis Function ("RBF"):

$$\begin{cases} \phi_{poly}(X_i, X_j) = (X_i X_j)^d \\ \phi_{rbf}(X_i, X_j) = \exp\left(-\gamma ||X_i - X_j||^2\right) \end{cases} \tag{13}$$

where d is the order of the polynomial kernel and γ is a parameter related to the inverse of the variance of the Gaussian RBF.

The main advantage of SVM over LR is that SVM enables the generation of nonlinear classification regions [51]. However, determining whether it is better to use a polynomial kernel function, an RBF kernel function or a linear kernel function in a specific case is not trivial. It is also not trivial to select the best values of the polynomial order d, γ, and the regularization constant C. Therefore, these parameters should be regarded as hyperparameters, and the combination of hyperparameters that yields the highest accuracy must be determined. This significantly increases the computational cost. Table 5 illustrates the intervals between which the hyperparameters were optimized to find the maximum prediction accuracy.

4.5.4 Tree-based ensembles

Decision tree-based algorithms automatically find a set of if-then rules whose final output is the class to be predicted [51]. These models are built recursively by making binary partitions to grow and create the branches of the decision tree. To speed up the calculation of the variables on which the partition is selected, random sampling is performed. To select the sample size, a hyperparameter named max features (Max_f) is proposed and is usually adjusted as the integer value of the square root of the number of variables.

To avoid overfitting the data, the decision tree hyperparameters are the maximum depth of the tree l, the minimum number of samples in a leaf (Min_{sl}), and the minimum number of samples in a split (Min_{ss}). Nonetheless, decision trees tend to overfit the training data. The sensitivity of decision trees to small changes in the values of the training data generates abrupt changes in tree construction [51]. These disadvantages are overcome by using the RF and GB algorithms, which use different ensemble techniques to calculate predictive models.

Random forests

RF is an ensemble learning method for classification, regression, and other tasks that operates by constructing a multitude of decision trees at training [51]. The method used in RF is bootstrap aggregation (or bagging), which consists of taking K repeated samples of the training dataset and creating several $\hat{f}^{*k}(X)$ models based on the sample. In this way, the output of the final model is the average of all outputs for all models, as indicated by Eq. (14):

$$\hat{f}_{bag}(X) = \frac{1}{K}\sum_{k=1}^{K}\hat{f}^{*k}(X) \tag{14}$$

where K is the bootstrap aggregation number of estimators and $\hat{f}^{*k}(X)$ are decision trees, which were trained by randomly using a subset of input variables to build them. A large value of K increases the computational cost of training, but a very small number can lead to low model precision. Since it is not possible to know what the best value of K is, it must be treated as one more hyperparameter.

It should be noted that RF additionally shares the same hyperparameters as decision trees. Consequently, the computational cost of finding the best model requires additional permutations in the hyperparameters.

Gradient-Boosted Trees

Similar to RF, GB is an ensemble learning method for classification that trains a multitude of decision trees. However, GB employs a technique called boosting, where each tree is adjusted using a modified dataset based on the training residues r [51].

In each iteration, a new tree $\hat{f}^{*k}(X)$ is created with a training data set (X, r) instead of (X, y); the former data set gives more importance to misclassified observations in the previous tree. Then, at each boosting iteration, both the estimator function and the modified dataset based on the residuals r are updated, as shown in Eq. (15):

$$\begin{cases} \hat{f}_{boost}(X) \leftarrow \hat{f}_{boost}(X) + \lambda\hat{f}^{*k}(X) \\ r \leftarrow r - \lambda\hat{f}^{*k}(X) \end{cases} \tag{15}$$

where λ is a hyperparameter that adjusts the learning rate of the model. When all trees have been built, the boosting model is given by Eq. (16):

$$\hat{f}_{boost}(X) = \sum_{k=1}^{K}\lambda\hat{f}^{*k}(X) \tag{16}$$

GB has an additional hyperparameter named ***Subsample***, which indicates the fraction of samples used to train the decision trees. Smaller fractions reduce the algorithm's precision but increase the generalizations that the model can make about the training set. Last, as in RF, the maximum depth of the decision trees, the minimum numbers of samples in the leaves and in the splits, and the partition criteria must be adjusted.

4.5.5 Multilayer perceptron artificial neural network

An MLP neural network consists of an input layer, several hidden layers, and an output layer. The input layer is composed of the feature vector X. The series of hidden layers are each composed of a set of neurons, which are forward connected [51]. Equation 17 describes the mathematical model of a single hidden layer.

$$f(X) = \beta_0 + \sum_{k=1}^{K}\beta_k A_k(X) \tag{17}$$

where K is the number of neurons in the hidden layer and $A_k(X)$ is the activation of the neuron. $A_k(X)$ is described in Eq. (18).

$$A_k(X) = g\left(w_{k0} + \sum_{j=1}^{p} w_{kj}X_j\right) \tag{18}$$

$A_k(X)$ depends on the inputs of the previous layer X, an activation function g, and the connection weights among neurons w_{kj}. The activation function g is nonlinear, and values between the sigmoid function and the rectified linear unit (ReLU) function are usually chosen:

$$\begin{cases} \text{Sigmoid function: } g(z) = \dfrac{1}{1+e^{-z}} \\ \text{ReLU function: } g(z) = \max(0, z) \end{cases} \tag{19}$$

In addition, the weight w_k of each of the K neurons is estimated by minimizing the negative multinomial loglikelihood problem presented in Eq. (20) using the training observations (X_i, y_i) for $i = 1, ..., n$.

$$\min_{w} \sum_{i=1}^{n} \left[-Y_i \log(f(X_i)) - (1 - Y_i) \log(1 - f(X_i)) \right] \quad (20)$$

This optimization problem is solved iteratively using the descending gradient method, which adjusts the weights toward the opposite direction of the gradient of the error function, as shown in Eq. (21):

$$w \leftarrow w - \lambda \left(\frac{\partial L}{\partial w} \right) \quad (21)$$

where λ is the learning rate, which must be adjusted as another hyperparameter of the model. The gradient descent function can be modified to include the momentum hyperparameter β, which allows getting stuck in local minima solutions.

In general, MLP is capable of learning very complex nonlinear models. However, like ensemble models, there are many hyperparameters to tune: the number of neurons K in the hidden layer, the activation function g, and the learning rate λ.

4.5.6 Recurrent neural networks

RNNs, unlike MLP, can have connections to both layers ahead and the same layer [51]. The recurrent networks are trained using data sequences $X = X_1, \ldots, X_L$, where L is the length of the sequence. Equation (22) describes the mathematical model of a recurrent hidden layer:

$$A_{lk}(X) = g \left(w_{k0} + \sum_{j=1}^{p} w_{kj} X_{lj} + \sum_{s=1}^{K} u_{ks} A_{l-1,s} \right) \quad (22)$$

$A_k(X)$ depends not only on the inputs of the previous layer X_{lj} but also on the previous activation of the same-layer neuron $A_{l-1,s}$.

The gradient-based learning algorithm in standard RNNs such as in Eq. (22) gives rise to the well-known vanishing gradient problem, where the recursive model cannot capture the long-term interactions in the input sequence. To address this problem, LSTM networks, which allow the gradient to flow over large sequences during training, have been proposed. LSTM-type neurons have other additional inputs called gates, which allow neurons to regulate the flow of information that enters and is memorized. LSTM models are specifically designed to capture long-range temporal dependencies by preserving information across time steps through gated mechanisms. In the context of stock market forecasting, this allows the model to retain a memory of historical prices, whose corresponding trends or anomalies might influence future prices [53].

The network architecture proposed for this model consists of an LSTM layer of networks with K LSTM units. To regularize the network, a dropout layer is included after the LSTM layer, where the R_D hyperparameter adjusts the dropout rate.

The dropout layer is followed by a dense layer that is completely connected forward. In the dense layer, the hyperparameter g is increased to adjust the type of activation function. Similar to MLP, the λ parameter adjusts the learning rate.

4.5.7 Hyper-parameters optimization

As stated in the previous sections, hyperparameter tuning is a computationally intensive task, particularly in the GB algorithm, where the number of combinations of discrete and continuous parameters is huge. To address the hyperparameter calculation problem, two strategies are implemented. The first is a random search for combinations, rather than an exhaustive search for all combinations. To do this, a hyperparameter search is performed on 100 random combinations for each algorithm. The second is to train the machine learning algorithm on a random subset of the training data. This speeds up the search for hyperparameters, but may not yield the model with the best performance. Table 5 summarizes the hyperparameters that were fit and the ranges for the random search.

Because all algorithms are trained and tested with the same data, and because all algorithms have been subject to the same hyperparameter optimization process, each algorithm is given an equal opportunity to achieve the highest possible accuracy. The following subsection explains how algorithm accuracy is analyzed.

4.6 Analysis of algorithm accuracy

The literature mentioned in Sect. 3 compares different algorithms on the basis on their average accuracy rates. While average accuracy rates provide useful information, we start from the premise according to which average accuracy rates may not be sufficient to rank different machine algorithms according to their respective levels of accuracy. Therefore, we created a set of four nonparametric measures that complement the analysis of average values to have a clearer picture of the ability of algorithms to accurately predict stock market prices. Because these measures are nonparametric, they have the advantage of not requiring assumptions about accuracy rates and their distribution.

We employed four nonparametric measures to compare the relative accuracy of algorithms: ordinal ranking of the algorithms according to their respective accuracy for each period-market pair, highest accuracy throughout time in a

Table 5 Hyperparameter values

Algorithm	Symbol	Hyperparameter	Values
SVM	C	Regularization constant	[0.1, 100]
	γ	Kernel coefficient	[0.0001, 1]
	d	Polynomial Degree	[2,5]
	ϕ	Kernel Function	[Rbf, Linear, Poly]
RF	l	Max depth	[2, 10]
	Max_f	Max features	[log2, sqrt]
	Min_{sl}	Min samples by leaf	[2, 5]
	Min_{ss}	Min samples by split	[2, 10]
	K	Number of estimators	[100, 1000]
GB	λ	Learning Rate	[0.01, 0.1]
	l	Max depth	[2, 10]
	Max_f	Max features	[log2, sqrt]
	Min_{sl}	Min samples by leaf	[2, 10]
	Min_{ss}	Min samples by split	[2, 5]
	K	Number of estimators	[100, 1000]
	SS	Subsample	[0,1]
ANN	K	Neurons in the hidden layer	[1, 100]
	g	Activation Function	[Tanh, ReLU, Logistic]
	λ	Learning rate	[0.0001, 0.1]
	β	Momentum	[0.0001, 1]
LSTM	K	Number of units	[1, 100]
	R_D	Dropout rate	[0, 0.5]
	g	Activation Function	[Sigmoid, ReLU]
	λ	Learning rate	[0.0001, 0.1]

given market, the required number of periods to reach the highest accuracy, and the difference between highest accuracy and accuracy in the last period. An algorithm was considered optimal if it (i) had a higher ranking than the other algorithms; (ii) had a greater highest accuracy than the other algorithms; (iii) reached its highest accuracy more quickly than the other algorithms; and (iv) had the smallest difference between highest accuracy and final accuracy. We analyzed which algorithm possessed the best trade-off among these four dimensions, which were second-best solutions, and which had the worst performance. The employed nonparametric measures arise from the identification of ordinal rankings and specific time patterns that enable a more precise assessment of the accuracy rates achieved by various algorithms.

To analyze the algorithms in relation to market efficiency, we relied on dynamic panel data analysis, as our data set consists of a large number of stock markets (55) and a large number of periods (65 two-month periods). Dynamic panel data analysis explicitly considers both the time dimension of the data set and the specificity of each market, thus enabling the detection of statistically significant relationships that cannot be detected by analyses of pure time-series or pure cross-sectional data [54]. For our data set, the dynamic panel data analysis started from the hypothesis that the accuracy of an algorithm can be explained by its past values in addition to the current and past values of market efficiency. Specifically, assume that the accuracy of an algorithm is better explained by the combination of its past values with past efficiency values than by its past values alone. In this situation, market efficiency plays a role in algorithm accuracy. This provides an adequate characterization of the complex nature of stock market prediction.

5 Results

We first compare the relative accuracy of each algorithm (5.1) and then analyze algorithm accuracy in relation to market efficiency (5.2).

5.1 Accuracy of machine learning models

Table 6 provides descriptive statistics for the accuracy of each algorithm. Interestingly, the shape statistics suggest that accuracy tends to be normally distributed for all algorithms, thus making them easier to compare. Indeed, skewness and Pearson's second coefficient of skewness, which measure distribution symmetry, are near zero for all algorithms, consistent with normal distributions. Similarly, kurtosis, which measures the concentration of values around the mean, tends to be close to 3. Ultimately, the shares of values concentrated within one standard deviation of the mean and two standard deviations of the mean are approximately the same as the shares associated with a normal distribution (68% and 95%, respectively). The relative match with a normal distribution is confirmed in Fig. 5, where the black curve is the kernel density estimation of the corresponding frequency distribution and the red curve is a normal distribution whose mean and standard deviation correspond to the accuracy of the given algorithm. For instance, the red curve associated with SVM is a normal distribution with a mean of 0.5164 and a standard deviation of 0.0865. Figure 5 shows that the two curves are largely superposed.

Addressing our first research question (see Sect. 3) *of whether a particular algorithm exhibits superior predictive accuracy*, Table 6 suggests that the most accurate algorithm is SVM, which has the highest mean accuracy (0.5164). Therefore, *SVM is the best stock market predictor on average according to our sample* and should therefore be the first choice for trading strategies—either manual trading strategies or algorithm trading strategies—when forecasts about stock indexes are required. ANN and GB have the second-best performance, whereas RF, NB, LR, and LSTM have the worst performance.[3]

Notwithstanding, average accuracy is approximately 51% for all algorithms. Put differently, as our analysis rests upon a large sample of markets and a large number of time periods, algorithms are able to predict only about half of stock index price movements on average. This suggests that predicting stock market prices on the basis of technical indicators is a difficult task for the algorithms under consideration. Consequently, further research should inquire into the improvement of algorithm accuracy using data that is complementary to technical indicators. As explained before, technical indicators have the advantage of being inferred from market prices, which are widely available and objective data. Nevertheless, the low average accuracy rate suggests that technical indicators, despite being necessary for stock market prediction, are not sufficient. Complementary data could arise from "fundamental" variables, namely variables arising from the financial statements published by the companies listed in each stock market. Among these variables, we can cite earnings per share (EPS), price-to-earnings ratio (P/E), debt-to-equity ratio (D/E) and return on equity (ROE). Combining technical indicators with fundamental analysis for the purpose of stock market prediction remains relatively underexplored [1, 55], and the present paper confirms that this kind of combination is relevant for future research. Another solution could be text mining based on financial news associated with listed companies, and the corresponding recognition and classification of investors' emotional tendencies [56]. Furthermore, macroeconomic indicators could be employed as data complementary to technical indicators, because macroeconomic indicators drive broader market sentiment, which in turn influences the price of stock indices [57–60].

As mentioned in Sect. 3, some studies suggest that the accuracy rate of algorithms can reach 90%. However, these studies are focused on one or a small sample of stock indices. Our analysis shows that, over a large number of stock indices (and a large number of periods), an accuracy rate of 90% is rather the exception than the rule because the average accuracy rate across markets is approximately equal to 51%. Even the third quartile is not greater than 58%, as described in Table 6. Focusing on a small number of markets could be misleading as the corresponding accuracy rate is not sufficiently representative. Figure 4 presents the average precision depending on the markets and algorithms. The graph shows that although the accuracy of an algorithm varies across markets, these variations are usually smaller than those reported in the literature. The algorithm with the best performance on average is SVM for the **BSE DC** stock index, with a value of 60.37%, and the worst-performing algorithm on average is NB for the **Moroccan AS** stock index, with a value of 47.78%.

A simple average across markets and time might not be precise enough to assess algorithm accuracy. In addition, the average accuracies of SVM, ANN, and GB are rather similar. Therefore, a simple average does not indicate a clear-cut advantage of SVM. To obtain a clearer picture of algorithm accuracy, other tools are necessary. The left graph in Fig. 5 presents the ordinal ranking of the algorithms according to their accuracy on each market and each period separately. An algorithm is ranked first if its accuracy is highest for a given market-period pair, second if its accuracy is second highest, and so on until the last (seventh) rank for lowest accuracy. Formally, let $R^a_{i,t}$ denote the rank of algorithm a for market i in period t and $Y^a_{i,t}$ denote the accuracy associated with (a, i, t). Therefore,

$$R^a_{i,t} = \#\{b \neq a \mid Y^b_{i,t} > Y^a_{i,t}\} + 1 \tag{23}$$

If two algorithms have the same accuracy for a given market-bimester pair, they have the same rank. Each box is associated with an algorithm and a market, and represents the average ranking of that algorithm across periods, namely $\frac{1}{T}\sum_{t=1}^{T} R^a_{i,t}$, where each period is identified by a positive integer running from 1 to T.

SVM has the highest "global" mean rank, that is, the mean rank across bimesters and then across markets. Consequently,

[3] In the case of ensemble algorithms, namely RF and GB, training a multitude of decision trees was therefore not able to generate the best average accuracy in our study.

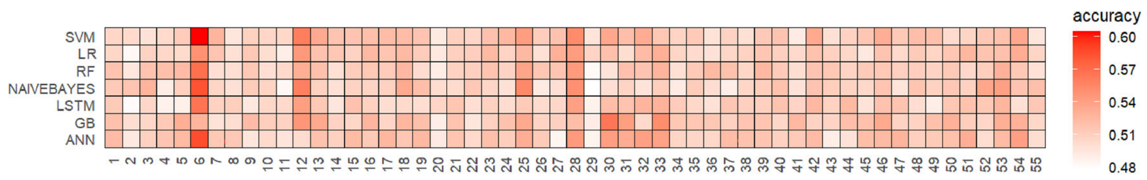

Fig. 4 Average accuracy comparison for all algorithms and the market sample detailed in Table 2. *Source*: Authors' calculations. Each number corresponds to stock market. Table 2 provides details about the name and location of each stock market

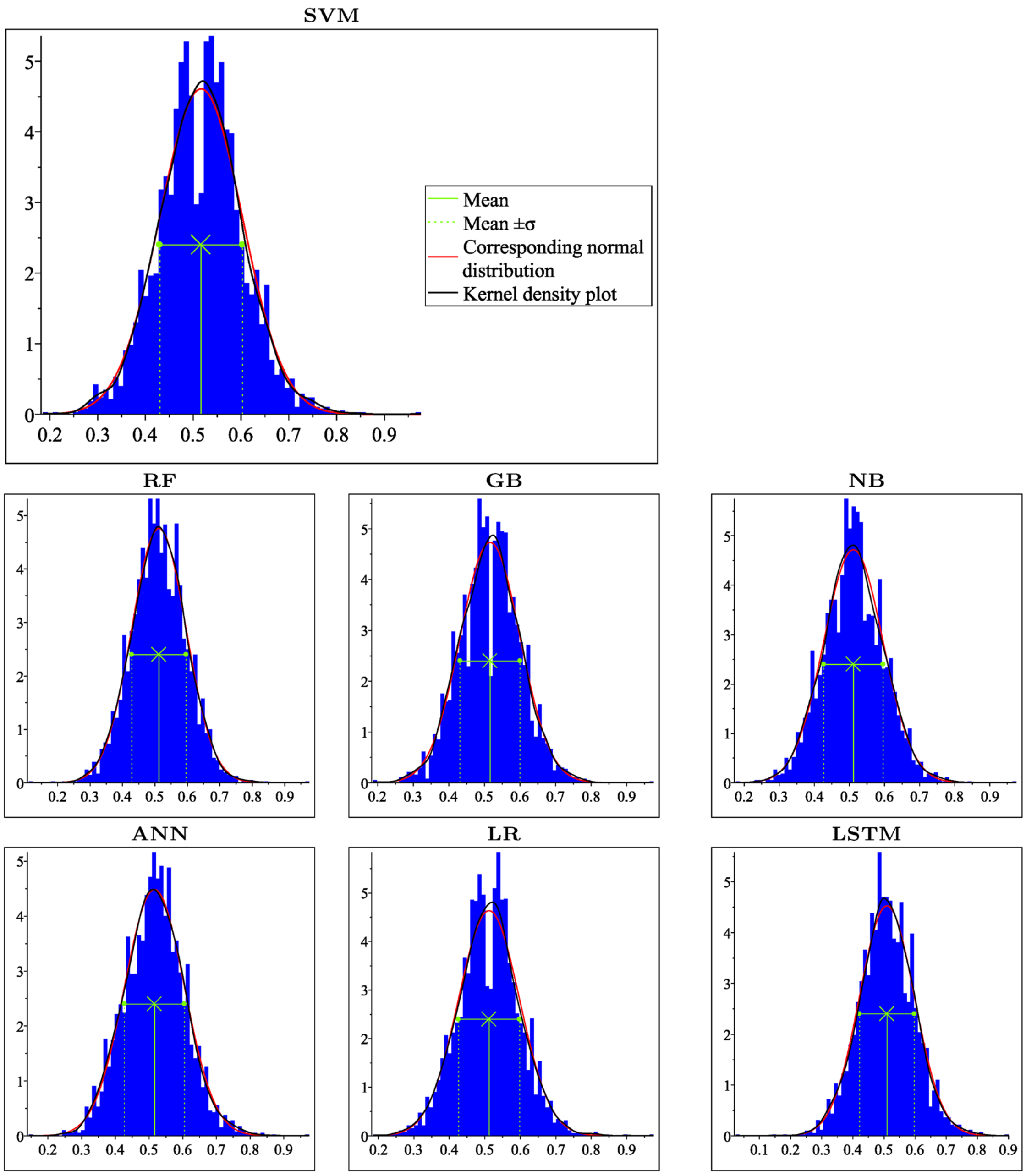

Fig. 5 Frequency distributions of algorithm accuracy. *Source*: Authors' calculations. Note: Each histogram is rescaled to make the total area of the bars equal to 1. Kernel density plots are created using the standard Gaussian kernel

Table 6 Descriptive statistics of algorithm accuracy

	Mean	S.D	Min	Q1	Median	Q3	Max
SVM	0.5164	0.0865	0.1852	0.4634	0.5128	0.5714	0.9756
RF	0.5122	0.0838	0.1111	0.4545	0.5122	0.5682	0.9756
GB	0.5161	0.0843	0.1875	0.4615	0.5128	0.5714	0.9756
NB	0.5114	0.0846	0.1765	0.4545	0.5116	0.5682	0.9756
ANN	0.5160	0.0889	0.1429	0.4545	0.5122	0.5750	0.9756
LR	0.5117	0.0860	0.1852	0.4545	0.5116	0.5650	0.9756
LSTM	0.5095	0.0881	0.0244	0.4524	0.5116	0.5682	0.9000

	Skewness	Pearson's Skewness	Kurtosis	Concentration within one S.D. of the mean	Concentration within two S.D. of the mean
SVM	0.0884	0.1242	3.4468	0.6959	0.9519
RF	0.0242	0.0012	3.4410	0.7035	0.9552
GB	0.0317	0.1179	3.3505	0.6901	0.9592
NB	0.0987	−0.0087	3.3576	0.6929	0.9538
ANN	0.1260	0.1298	3.3518	0.6959	0.9561
LR	0.0939	0.0041	3.5348	0.6903	0.9564
LSTM	0.0004	− 0.0713	3.5586	0.6959	0.9561

Source: Authors' calculations

the market-specific ranking in the left graph in Fig. 6 confirms that SVM tends to give better predictions than the other algorithms. SVM is followed by GB and ANN. Interestingly, the first three algorithms according to the previous ordinal ranks—SVM, GB, and ANN—are the same as those arising from the simple arithmetic means in Table 6.

The right graph in Fig. 6 describes the learning abilities of each algorithm. Learning abilities are captured by three criteria. The first criterion, the number of bimesters an algorithm requires to reach its maximum accuracy for a given market, is reported on the horizontal axis. This number is conceptualized as the "speed" of learning by an algorithm. A higher speed implies that an algorithm learns to predict stock market prices more quickly, which is a valuable criterion for comparing algorithm accuracy. The second criterion, reported on the vertical axis, is the difference between the accuracy in the final bimester and the maximum accuracy reached in the past. This loss of accuracy from maximum accuracy to final accuracy is conceptualized as the "depreciation" of an algorithm's learning ability. Greater depreciation implies that an algorithm is less able to maintain its accuracy around the highest accuracy achieved in the past. Last, for each speed-depreciation pair, the graph indicates the mean highest accuracy, which represents the "magnitude" of an algorithm's learning ability. Ultimately, an algorithm is better for stock market prediction if the corresponding trade-off among learning speed, depreciation, and magnitude is better. According to that trade-off, SVM remains the most accurate of the algorithms. Indeed, SVM ranks third in speed, fifth in deprecation, and second in mean highest accuracy. This gives rise to an average rank equal to 3.33. No other algorithm achieves such a rank. GB, LR, and LSTM have average ranks of 3.67 and are followed by RF (4.00), ANN (4.33), and NB[4] (5.33). The superiority of SVM is interesting because SVM is not the most complex algorithm. Our results confirm that, as already suggested by [6, 7], baseline algorithms like SVM may perform better than ensemble algorithms and deep learning algorithms for stock market prediction. Nevertheless, being a baseline algorithm does not guarantee better accuracy. For example, LR shows the highest depreciation, NB shows the lowest speed, and the average accuracy of both algorithms is among the lowest (only LSTM has a lower average accuracy than LR and NB).

Before getting to the influence of market efficiency on algorithm accuracy regarding stock market prediction, we summarize our key findings about algorithm accuracy as follows. On the basis of our large sample of markets and a large number of time periods:

1. Algorithms reach an average accuracy rate of approximately 51%.

[4] If we compare GB with RF, namely the two ensemble algorithms of our study, we can observe that GB has a higher average accuracy than RF (see Table 6). In addition, its mean highest accuracy (see MHA in the right side of Fig. 6) is greater. However, RF possesses higher speed and lower depreciation (see the right side of Fig. 6). Therefore, GB performs better than RF according to two criteria, but RF performs better than GB according to two other criteria. Hence, one may infer that both algorithms share rather similar accuracy, yet GB has a slight advantage arising from market-specific ordinal ranking, which is the fifth criterion of our analysis (see the left side of Fig. 6).

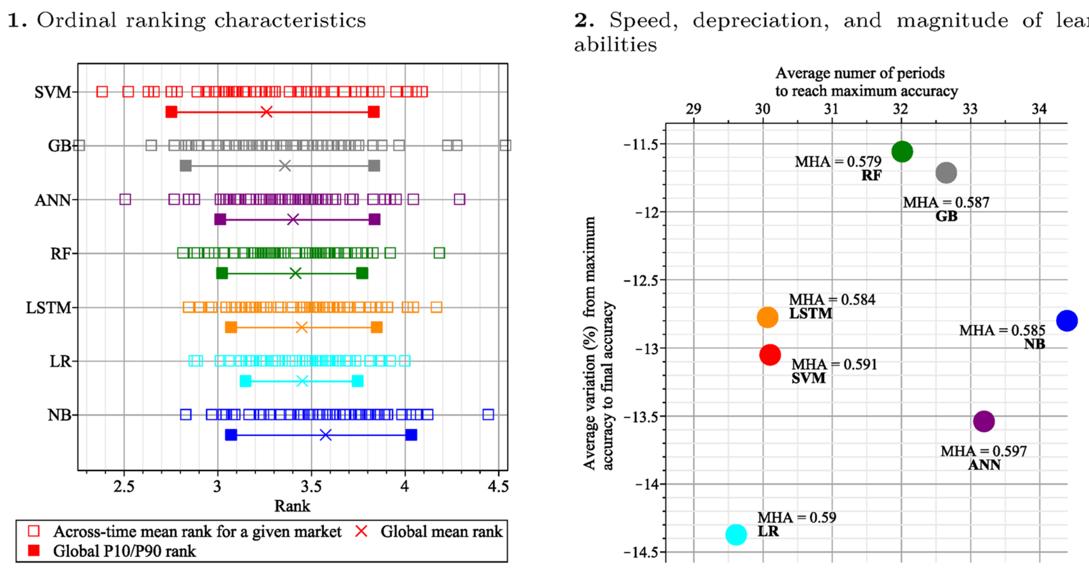

Fig. 6 Nonparametric measures of algorithm accuracy. *Source*: Authors' calculations. On the left, each box corresponds to the across-time mean ranking of algorithms according to their accuracy in each market. The cross is the global mean rank (i.e., across time and then across-average mean rank for each market) and the solid boxes are the 10th and 90th percentiles. On the right, the learning "speed" of an algorithm is the number of two-month periods required for an algorithm to reach its maximum accuracy on a given market (x-axis). Depreciation is the decrease in algorithm accuracy from the bimester with the highest accuracy to the final bimester (y-axis). Last, MHA is the across-market mean value of the highest accuracy. Calculations are based on 12-month moving averages to lessen sensitivity to periodic fluctuations in algorithm accuracy

2. Support vector machines (SVM), despite not being the most complex algorithm, is the best stock market predictor on average.
3. ANN and GB have second-best average accuracy rates, whereas RF, NB, LR, and LSTM have the worst average accuracy rates.
4. Ordinal rankings and other nonparametric measures in terms of "speed", "depreciation" and "magnitude" confirm the superiority of SVM in terms of accuracy.
5. Ultimately, the superiority of SVM in terms of accuracy is robust to a full set of measures.

Note that these results arise from the use of technical indicators as input data without complementary data based on financial statements or financial news, as discussed before. Precisely, maybe an algorithm different from SVM would possess higher accuracy with complementary, and this alternative algorithm could be an ensemble algorithm or a deep learning algorithm instead of a baseline algorithm like SVM.

5.2 Algorithm accuracy and market efficiency: dynamic panel data analysis

Table 4 shows that average accuracy is heterogeneous across markets. Does market efficiency explain this heterogeneity? Put differently, does accuracy improve if market efficiency is lower? Precisely, this kind of question is our second research question (see Sect. 3).

Surprisingly, our results show that the relationship between algorithm accuracy and market efficiency, if it exists, is very weak. Table 7 provides descriptive statistics for market efficiency in our sample, and Fig. 7 presents the corresponding frequency distributions. Similar to algorithm accuracy, market efficiency tends to be normally distributed. Nevertheless, the scatter plots in Figs. 8 and 9 show that there is not a straightforward relationship between stock market efficiency and algorithm accuracy. The corresponding sets of efficiency-accuracy points do not fit well into a simple linear regression. This is also confirmed by the correlation matrix in Table 8. The coefficients of correlation are relatively close to zero and are positive for all algorithms when the efficiency metric is the Hurst exponent and negative when the efficiency metric is HW or RG (except the coefficient associated with the pair HW-GB). In accordance with Table 4 and with the slopes of the simple linear fits in Figs. 8 and 9, these signs imply that algorithm accuracy is negatively correlated with stock market efficiency. However, the highest coefficient of correlation in absolute terms is 0.0686 (algorithm NB and efficiency metric HW), suggesting a weak correlation.

Consequently, market efficiency does not seem to significantly influence algorithm accuracy. Notwithstanding, to better inquire into the relationship between efficiency and algorithm accuracy, a model other than standard linear regres-

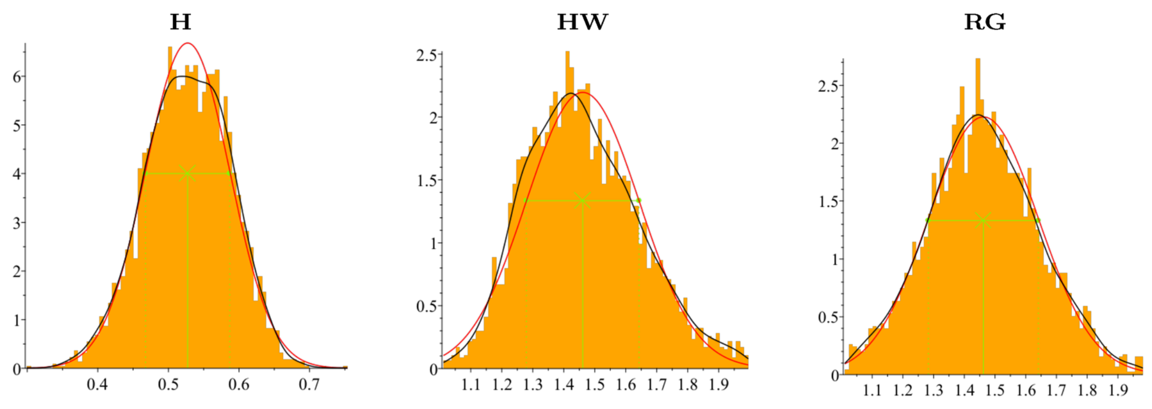

Fig. 7 Frequency distributions of stock market efficiency. *Source*: Authors' calculations. The black curve is the Gaussian kernel density estimation of the corresponding frequency distribution. The red curve is a normal distribution whose mean and standard deviation are those of the given variable. The green solid line is the mean, and the green dotted lines represent one standard deviation of the mean

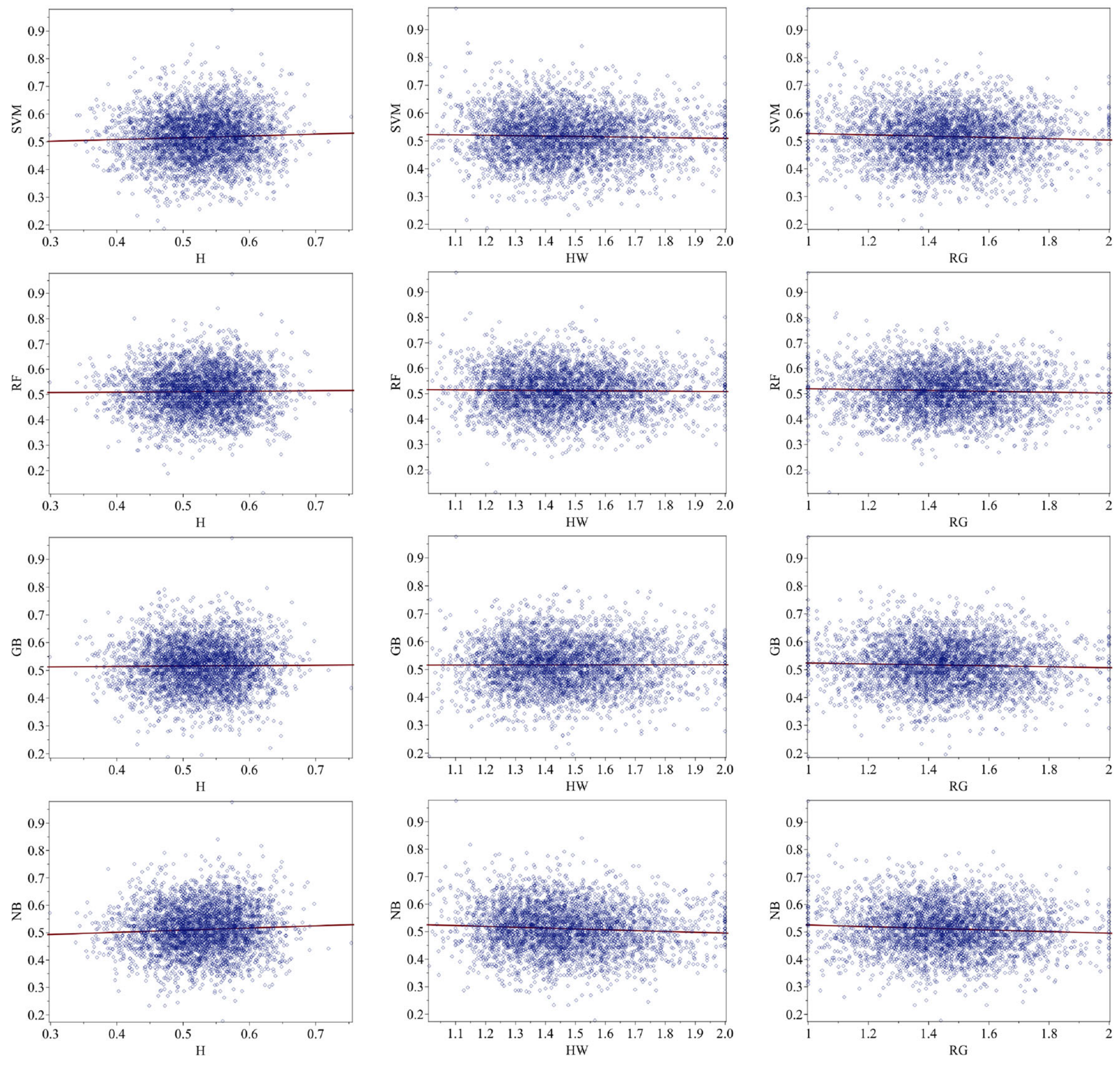

Fig. 8 Scatter plots for each pair of efficiency metric (x-axis) and algorithm accuracy (y-axis) (1). *Source*: Authors' calculations. Note: In each graph, the line corresponds to a simple linear fit of the data by ordinary least squares

Table 7 Descriptive statistics of market efficiency

	Mean	S.D	Min	Q1	Median	Q3	Max
H	0.5272	0.0597	0.2992	0.4859	0.5281	0.5701	0.7539
HW	1.4613	0.1817	1.0000	1.3269	1.4460	1.5812	2.0000
RG	1.4621	0.1791	1.0000	1.3406	1.4563	1.5814	2.0000

	Skewness	Pearson's Skewness	Kurtosis	Concentration within one S.D. of the mean	Concentration within two S.D. of the mean
H	−0.1239	− 0.0430	2.8042	0.6765	0.9572
HW	0.3764	0.2528	2.8279	0.6711	0.9564
RG	0.1141	0.0971	2.8094	0.6843	0.9526

Source: Authors' calculations

Table 8 Correlation matrix between algorithm accuracy and market efficiency

	SVM	RF	GB	NB	ANN	LR	LSTM
H	0.0446	0.0126	0.0108	0.0555	0.0313	0.0319	0.0003
HW	−0.0306	−0.0165	0.0019	−0.0686	−0.0297	−0.0334	−0.0143
RG	−0.0514	−0.0380	−0.0377	−0.0671	−0.0379	−0.0430	−0.0301

Source Authors' calculations

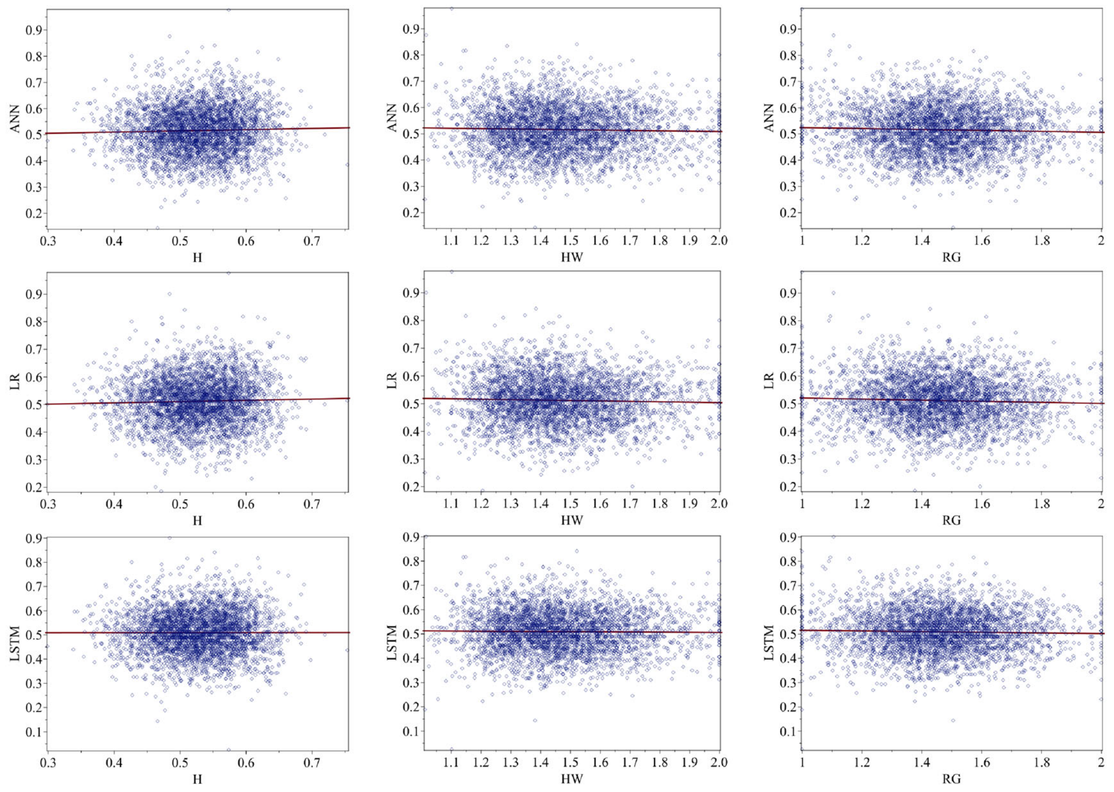

Fig. 9 Scatter plots for each pair of efficiency metric (x-axis) and algorithm accuracy (y-axis). *Source*: Authors' calculations. Note: In each graph, the line corresponds to a simple linear fit of the data by ordinary least squares

sion is required. In this regard, we use the autoregressive distributed lag (ARDL) model. Let $X_{i,t} \in [a, b] \subset \mathbb{R}_+$ be an efficiency measure (e.g., the RG metric, which ranges from 1 to 2) and $Y_{i,t} \in [0, 1]$ the accuracy rate of an algorithm (e.g., SVM) for the stock index in market i among some set J of markets and some two-month period $t = 1, 2, \dots, T$ (two-month period 1 is 2011/01-02, two-month period 2 is 2011/03-04, and so on until period $T = 65$, which is 2022/06-07). The ARDL model is applied to the panel data set $\{\langle X_{i,t}, Y_{i,t}\rangle \mid i \in J \text{ and } t = 1, 2, \dots T\}$ arising from each efficiency measure-accuracy rate pair associated with a given value of (i, t). The model is as follows:

$$Y_{i,t} = \sum_{u=1}^{p} \lambda_u Y_{i,t-u} + \sum_{u=0}^{p} \beta_u X_{t-u} + \mu_i + \varepsilon_{i,t} \tag{24}$$

For each market i and period t, variable Y is conceptualized as a linear combination of its p past values in $t-1, t-2, \dots, t-p$ and the p past values of variable X in $t-1, t-2, \dots, t-p$.[5] The aforementioned past values is what makes the analysis dynamic. Contrary to traditional static panel data analysis, dynamic panel data analysis has the following advantage: past values of the dependent variable capture the determinants of algorithm accuracy different from market efficiency. Consequently, a sufficient number of lags avoids omitted variable bias. Thereafter, remaining endogeneity concerns are addressed by a two-step estimation of Eq. (24). First, Eq. (24) is estimated according to ordinary least squares (OLS). Thereafter, following [61] and [62], the OLS estimates are adjusted by the half-panel jackknife (HPJ) method from [63] to be unbiased estimates. Variable μ_i is a market-specific time-constant term that captures market heterogeneity arising from differences in traded volumes, volatility, liquidity, and financial reporting, among other characteristics of stock markets. Last, variable $\varepsilon_{i,t}$ is a market-specific i.i.d. error term with zero mean and finite variance. Variable $\varepsilon_{i,t}$ embodies the factors that influence Y and are different from both X and the stock market characteristics associated with μ_i.

Assume that the estimation of Eq. (24) leads to accepting the hypothesis $\beta_{i,u} = 0\ \forall(i, u)$. In that case, Eq. (24) would read $Y_{i,t} = \sum_{u=1}^{p} \lambda_u Y_{i,t-u} + \mu_i + \varepsilon_{i,t}$. Therefore, the algorithm's accuracy would be better explained by the past values of itself *independently of* market efficiency, similar to a standard autoregressive process without market efficiency as an exogenous regressor. According to the statistical concept of causality [64–66], market efficiency would not *Granger-cause* algorithm accuracy. We implement this Granger no-causality test to confirm if market efficiency influences algorithm accuracy. The null hypothesis is $\beta_u = 0$ for all values of u. The null is tested for each algorithm and each measure of market efficiency. We accept the null hypothesis if the corresponding Wald Chi-squared statistic has a p-value greater than the 10% significance level (more details about this statistic are available in [67]).

Equation 24 assumes that parameters λ_u and β_u are common to all markets; μ_i remains specific to each market to account for market-specific unobserved heterogeneity. Pooled estimation of λ_u and β_u is more consistent with the large values of both N (number of markets) and T (number of periods) in our dataset. In addition, the Wald Chi-squared statistic underlying the Granger no-causality test is computed to be robust against the presence of heteroskedasticity of the error term along both the time and cross-sectional dimensions of the panel. Ultimately, the choice of p (number of lags) rests upon the standard minimization of the Bayesian information criterion (BIC). All calculations are based on the **xtgrangert** routine in Stata [67].

Table 9 presents the results of the Granger non-causality tests. The corresponding estimation of Eq. (24) is presented in Table 10. For each estimation, BIC minimization leads to the use of a single lag, namely, $p = 1$ in Eq. (24). The respective values of BIC are reported in Table 11. According to Table 9, only in the case of algorithms RF and LR do we fail to accept the null hypothesis of Granger non-causality when market efficiency is measured by the Hurst exponent (H). However, the two other measures of market efficiency—HW and RG—strongly support the acceptance of the null hypothesis. In addition, for all other algorithms, the null hypothesis of Granger non-causality is accepted independently of the measurement of market efficiency. Therefore, as a general result, we can state that algorithm accuracy is "immune" from market efficiency. The accuracy of algorithms for predicting market trends on the basis of technical indicators as shown in Table 3 is not influenced by market efficiency. Very interestingly, this result is robust to three different ways to quantify of market efficiency.

Theory suggests that market efficiency affects the accuracy of stock market predictions in general. Surprisingly, our dynamic panel data analysis suggests that machine learning algorithms applied to technical indicators depart from this negative relationship between market efficiency and stock market prediction. What can explain the fact that algorithm accuracy only reaches approximately 51%? It is possible that the sole use of technical indicators is not sufficient for accurate predictions, which warrants further research.

5.3 Hyperparameters

Statistical summaries of the hyperparameters selected as a result of the random search of hyperparameters are presented in Tables (12) and (13). Analyzing the hyperparameters of

[5] The number of lags of variable Y, p, is the same as the number of lags of variable X. This restriction has the advantage of significantly reducing the computational time required to estimate Eq. (24).

Table 9 Granger non-causality tests based on Eq. (24) for causal relationship from market efficiency to algorithm accuracy

		SVM	RF	GB	NB	ANN	LR	LSTM
H	Wald statistic	0.0106	2.7268	1.4005	2.3524	1.3551	3.2377	1.7843
	p-value	0.9180	0.0987*	0.2366	0.1251	0.2444	0.0720*	0.1816
HW	Wald statistic	0.0884	0.0026	2.1044	0.0077	0.3809	0.5703	0.0007
	p-value	0.7662	0.9595	0.1469	0.9300	0.5371	0.4501	0.9786
RG	Wald statistic	0.7967	1.4438	0.6175	1.8177	0.4813	0.8871	0.7810
	p-value	0.3721	0.2295	0.4320	0.1776	0.4878	0.3463	0.3768

Source Authors' calculations. *p-value $< 10\%$. The null hypothesis is Granger non-causality from market efficiency to algorithm accuracy, which is accounted by $\beta_{i,u} = \beta_u = 0 \; \forall (i, u)$ in Eq. (24)

Table 10 Half-Panel Jackknife estimation of Eq. (24)

		SVM	RF	GB	NB	ANN	LR	LSTM
H	β_{t-1}	0.0036	0.0427	−0.0321	0.0486	0.0302	0.0534	0.0448
	z-stat	0.09	1.83	−1.34	1.7	1.06	1.81	1.22
	p-value	0.925	0.068*	0.181	0.089	0.289	0.071*	0.224
HW	β_{t-1}	−0.0025	0.0004	0.0133	0.0007	−0.0053	−0.0052	0.0002
	z-stat	−0.31	0.05	1.38	0.09	−0.54	−0.67	0.03
	p-value	0.759	0.957	0.166	0.927	0.589	0.502	0.978
RG	β_{t-1}	0.0081	0.0109	0.0058	0.0107	0.0053	0.0090	0.0081
	z-stat	0.92	1.22	0.78	1.36	0.76	0.88	0.93
	p-value	0.357	0.222	0.438	0.173	0.448	0.38	0.353

Source: Authors' calculations. For each estimation, BIC minimization leads to the use of a single lag, i.e., $p = 1$ in Eq. (24). * p-value $< 10\%$

Table 11 Value of Bayesian Information Criterion for lag selection when estimating Eq. (24)

	SVM	RF	GB	NB	ANN	LR	LSTM
H	−16892.96	−16993.98	−16984.67	−17031.83	−16694.44	−16838.63	−16667.92
HW	−16857.78	−16978.16	−16998.96	−17006.14	−16686.21	−16811.98	−16625.41
RG	−16876.14	−17003.44	−16982.64	−17008.12	−16668.12	−16827.19	−16636.26

Source: Authors' calculations

SVM shows that most of the best models for different stock markets select low values of C and low values of γ. This indicates that, in general, values that do not overfit the training data are preferred. However, the most selected kernel function (ϕ) is the polynomial kernel with the highest order in the range provided and therefore, corresponds to the most complex polynomial model. Another interesting finding is the selection difference in the number of trees in the RF and GB algorithms (K). In the case of RF, a low value was preferred, while in the case of GB a high value was preferred. It should be noted that both had a significant standard deviation, indicating that there is no rule of thumb to define the number of trees. Furthermore, it was found that in the RF mostly shallow trees were created (l=2), while in GB they had a length twice that of the RFs (l=4).

Finally, for the GB, ANN, and LSTM algorithms, model accuracy is better at a low learning rate ($\lambda < 0.01$) than at a high rate. Hyperparameter tuning remains a challenging task and influences the accuracy of algorithms regarding stock market price forecasting. Poorly tuned hyperparameters can lead to underfitting or overfitting, and the same algorithm can behave very differently depending on hyperparameter values.

6 Conclusions and future work

The main objectives of this article were to determine which algorithm is most accurate for predicting stock indices and whether this accuracy is related to the efficiency of the index. To provide a robust conclusion, the accuracy of each of the algorithms was compared for multiple stock indices and for multiple time periods. Specifically, we performed a comparative analysis of different machine learning algorithms over a panel of 55 stock markets for a time span of 65 two-month periods from 2011/01 to 2022/07. In addition, we used dynamic panel data analysis to evaluate the significance of the relationship between market efficiency and algorithm accuracy. To measure market efficiency, we selected three

Table 12 Hyperparameter statistics—Numerical

Algorithm	Symbol	Mean	S.D.	Min	Median	Max
SVM	C	207.83	291.24	0.24	20.27	934.98
	γ	0.28	0.30	0.00	0.15	0.95
RF	K	422	241.75	103	343	961
GB	λ	0.02	0.02	0.0001	0.002	0.081
	K	612	294.48	125	642	996
	SS	0.40	0.33	0.10	0.32	1.00
ANN	K	41	30.09	4	29	95
	λ	0.01	0.02	0.0001	0.0008	0.078
	β	0.53	0.28	0.10	0.55	1.00
LSTM	K	50	22.03	12	48	97
	R_D	0.24	0.15	0.00	0.20	0.50
	λ	0.011	0.021	0.0001	0.01	0.10

Table 13 Hyperparameter statistics—Discrete

Algorithm	Symbol	Most Selected	Count
SVM	d	5	18
	Kernel Type	poly	23
RF	l	2	25
	Max_f	sqrt	30
	Min_{sl}	2	25
	Min_{ss}	2	12
GB	Min_{sl}	2	22
	Min_{ss}	9	13
	l	4	14
	Max_f	sqrt	29
ANN	g	ReLU	21
LSTM	g	ReLU	30

estimators (Hurst exponent, HW estimator, and RG estimator). Ultimately, SVM tended to exhibit the best average performance according to a comparative analysis based on a set of nonparametric measures of algorithm accuracy, with an accuracy of 51.64%. GB and ANN had the second-best average performance, with accuracies of 51.61% and 51.60%, respectively.

Although all algorithms had accuracies close to 50%, there was a high standard deviation of precision in the sample of at least 0.08. This shows that it is complex to model different stock markets in different periods. Surprisingly, we did not find an influence of market efficiency on algorithm accuracy. In dynamic panel data analysis, the Granger non-causality test based on the autoregressive distributed lag (ARDL) model shows that market efficiency did not influence algorithm accuracy. In addition, this result was robust to different measurements of market efficiency. In all estimations, the bulk of the Granger non-causality tests were associated with p-values high enough to prevent from rejecting the null hypothesis of Granger non-causality. Ultimately, these results suggest that the way algorithms use exclusively technical data from stock markets is immune to market efficiency, contrary to the theory of efficient markets.

Consequently, we propose increasing the number of model inputs in future work. All models were trained with the most commonly reported technical indicators. These technical indicators could be enriched with fundamental information related to each stock index and data extracted from news and social networks. It would be interesting to compare the predictive power of each algorithm with and without the enrichment of the model inputs to determine whether the technical indicators are sufficient to forecast stock prices.

Investors should not expect high accuracy rates on average from algorithms whose input comprises technical indicators exclusively. Technical indicators have the advantage of being easily available because they arise from stock prices, which are updated on a daily basis in various databases. However, to improve the accuracy rate, investors must be aware that technical indicators should be complemented by other sources of information, like text mining or data from financial statements. Combining technical indicators with other sources of information might be challenging, yet this is an indispensable task if investors want to better predict price movements and ultimately improve their gains as they trade stock indices in different markets. In addition, the efficiency metrics themselves could be part of the data that are employed as the input of algorithms; and maybe lower efficiency would improve algorithm accuracy as algorithms are fed with more complex data.

Finally, given that the methodology used to assess the accuracy of predictors included four nonparametric measures of equal importance, we hope in future work to incorporate a strategy for weighting each measure according to the characteristics of the problem to be predicted. In addition, an interesting path to explore would be the estimation of ARDL models with time-variant parameters in line with possible regime shifts regarding market efficiency. These regime shifts could be conceptualized based on patterns regarding volatility in stock markets in the aftermath of stock market shocks or economic crises [68]. Maybe this kind of model would improve our understanding of the impact of market efficiency on algorithm accuracy, and we could analyze algorithm accuracy in each regime. Indeed, maybe certain algorithms would perform better than others in periods of "high" volatility compared to periods of "low" volatility.

Appendix: Implementation details

In the first place, the historical data of the different stock indices mentioned in Table 2 were manually downloaded from www.investing.com. These indices were pre-processed by running a program created in R version 4.2.2 using the technical trading rules (TTR) library [43]. The efficiency estimators were calculated using the libraries Pracma [20] and Fractaldim [21]. The experiments were conducted on a local laptop equipped with 16GB of RAM and a 6-core Intel Core i7-10750 H processor.

Subsequently, a solution was created to train each of the machine learning models for each of the analyzed markets, depending on the number of hyperparameters and cross-validations. For the 5 algorithms that required hyperparameter optimization, the total number of models trained for the combinations mentioned above was 385,000 (10 k-folds, 100 hyperparameter combinations, 7 different models, and 55 different markets). To perform this hyperparameter selection over all markets, we implemented the Python class *model_selection.RandomizedSearchCV* to search the best models among all cross-validation partitions. To perform the random search, we used the following available classes on the scikit-learn library version 1.4.2: *svm.SVC, naive_bayes.GaussianNB, ensemble.RandomForestClassifier, ensemble.GradientBoostingClassifier, neural_network.MLP Classifier and linear_model.LogisticRegression* [69]. We also implemented the TensorFlow library [70] version 2.16.1 to train the LSTM models.

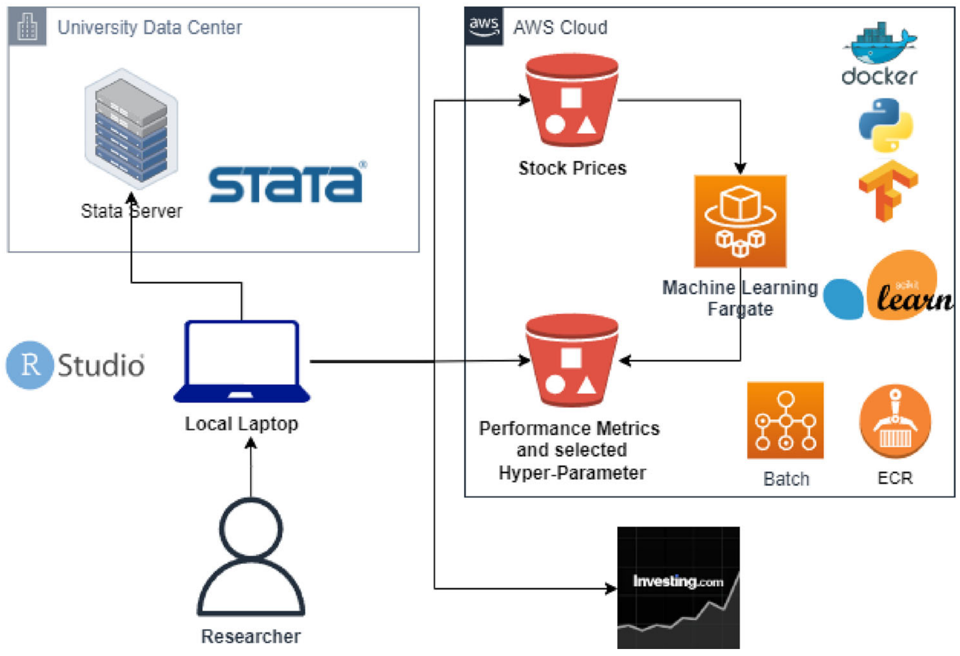

Fig. 10 Technological architecture

This implementation required a high-performance computing design in which the training of the various models was massively parallelized. Figure 10 describes the solution architecture implemented to train the machine learning models. The design strategy was to containerize the solution to be developed using dockerized Miniconda 3.0 image. Using AWS ECR, AWS Batch, and AWS Fargate tools, these images were run in the cloud in parallel for each of the algorithms and stock indices. The three AWS tools are serverless, so the cost is limited to the computing minutes used in the experiments. Both the information on technical indicators and the model training results were stored in CSV files in the AWS S3 storage service, which is also serverless. Finally, once the results were obtained, the econometric analysis was carried out on the StataBE 18 64-bit server provided by *Pontificia Universidad Javeriana*.

Author Contributions All authors (i) made substantial contributions to the final manuscript and (ii) read and approved the final manuscript.

Funding Open Access funding provided by Colombia Consortium. This work was supported by *Pontificia Universidad Javeriana*.

Data Availability Available on request to the corresponding author.

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no Conflict of interest.

Ethics approval not required since the present work did not involve human subjects and/or animals.

Consent to participate not required since the present work did not involve individuals.

Consent for publication not required since the present work did not involve individuals.

Code availability available on request to the corresponding author.

References

1. Lin, C.Y., Lobo Marques, J.A.: Stock market prediction using artificial intelligence: a systematic review of systematic reviews. Soc. Sci. Human. Open **9**, 100864 (2024). https://doi.org/10.1016/j.ssaho.2024.100864
2. Cavalcante, R.C., Brasileiro, R.C., Souza, V.L.F., Nobrega, J.P., Oliveira, A.L.I.: Computational intelligence and financial markets: a survey and future directions. Expert Syst. Appl. **55**, 194–211 (2016). https://doi.org/10.1016/j.eswa.2016.02.006
3. Kumbure, M.M., Lohrmann, C., Luukka, P., Porras, J.: Machine learning techniques and data for stock market forecasting: a literature review. Expert Syst. Appl. (2022). https://doi.org/10.1016/j.eswa.2022.116659
4. Kehinde, T.O., Chan, F.T.S., Chung, S.H.: Scientometric review and analysis of recent approaches to stock market forecasting: two decades survey. Expert Syst. Appl. (2023). https://doi.org/10.1016/j.eswa.2022.119299
5. Bustos, O., Pomares-Quimbaya, A.: Stock market movement forecast: a systematic review. Expert Syst. Appl. **156**, 113464 (2020). https://doi.org/10.1016/j.eswa.2020.113464
6. Tuekci, P.: Predicting the direction of movement for stock price index using machine learning methods. Adv. Intell. Syst. Comput. **427**, 477–492 (2016). https://doi.org/10.1007/978-3-319-29504-6_45
7. McCluskey, J., Liu, J.: US financial market forecasting using data classification with features from global markets. In: 2017 2nd International Conference on Image, Vision and Computing, ICIVC 2017, pp. 965–969. Institute of Electrical and Electronics Engineers Inc., New York (2017). https://doi.org/10.1109/ICIVC.2017.7984698
8. Fama, E.F.: Efficient capital markets: a review of theory and empirical work. J. Finance **25**(2), 383–417 (1970). https://doi.org/10.2307/2325486
9. Eom, C., Choi, S., Oh, G., Jung, W.-S.: Hurst exponent and prediction based on weak-form efficient market hypothesis of stock markets. Phys. A **387**(18), 4630–4636 (2008). https://doi.org/10.1016/j.physa.2008.03.035
10. Alvarez-Ramirez, J., Alvarez, J., Rodriguez, E., Fernandez-Anaya, G.: Time-varying Hurst exponent for US stock markets. Phys. A **387**(24), 6159–6169 (2008). https://doi.org/10.1016/j.physa.2008.06.056
11. Hall, P., Wood, A.: On the performance of box-counting estimators of fractal dimension. Biometrika **80**(1), 246–251 (1993). https://doi.org/10.2307/2336774
12. Genton, M.G.: Highly robust variogram estimation. Math. Geol. **30**(2), 213–221 (1998). https://doi.org/10.1023/A:1021728614555
13. Malkiel, B.G.: Efficient market hypothesis. In: Eatwell, J., Milgate, M., Newman, P. (eds.) The New Palgrave Dictionary of Finance, pp. 127–134. Palgrave Macmillan, London (1989). https://doi.org/10.1007/978-1-349-20213-3_13
14. Fama, E.F.: Random walks in stock market prices. Financ. Anal. J. **51**, 75–80 (1995)
15. Griffin, J.M., Kelly, P.J., Nardari, F.: Are emerging markets more profitable? Implications for comparing weak and semi-strong form efficiency. EFA 2007 Ljubljana Meetings Paper, AFA 2008 New Orleans Meetings Paper (2007) https://doi.org/10.2139/ssrn.959006
16. Puertas, A.M., Clara-Rahola, J., Snchez-Granero, M.A., De Las Nieves, F.J., Trinidad-Segovia, J.E.: A new look at financial markets efficiency from linear response theory. Financ. Res. Lett. **51**, 103455 (2023). https://doi.org/10.1016/j.frl.2022.103455

17. Di Matteo, T., Aste, T., Dacorogna, M.M.: Scaling behaviors in differently developed markets. Phys. A **324**(1–2), 183–188 (2003). https://doi.org/10.1016/S0378-4371(02)01996-9
18. Hurst, H.: Long term storage capacity of reservoirs. Trans. Am. Soc. Civ. Eng. (1951). https://doi.org/10.1061/TACEAT.0006518
19. Weron, R.: Estimating long-range dependence: finite sample properties and confidence intervals. Phys. A **312**(1–2), 285–299 (2002). https://doi.org/10.1016/S0378-4371(02)00961-5
20. Borchers, H.W.: Pracma: Practical Numerical Math Functions (2021). R package version 2.3.3. https://CRAN.R-project.org/package=pracma
21. Sevcikova, H., Percival, D., Gneiting, T.: Fractaldim: Estimation of Fractal Dimensions. (2014). R package version 0.8-4. https://CRAN.R-project.org/package=fractaldim
22. Shen, J., Shafiq, M.O.: Short-term stock market price trend prediction using a comprehensive deep learning system. J. Big Data (2020). https://doi.org/10.1186/s40537-020-00333-6
23. Yun, K.K., Yoon, S.W., Won, D.: Prediction of stock price direction using a hybrid ga-xgboost algorithm with a three-stage feature engineering process. Expert Syst. Appl. (2021). https://doi.org/10.1016/j.eswa.2021.115716
24. Kim, S., Ku, S., Chang, W., Chang, W., Chang, W., Song, J.W.: Predicting the direction of US stock prices using effective transfer entropy and machine learning techniques. IEEE Access **8**, 111660–111682 (2020). https://doi.org/10.1109/ACCESS.2020.3002174
25. Li, Y., Bu, H., Li, J., Wu, J.: The role of text-extracted investor sentiment in Chinese stock price prediction with the enhancement of deep learning. Int. J. Forecast. **36**, 1541–1562 (2020). https://doi.org/10.1016/j.ijforecast.2020.05.001
26. Hu, H., Tang, L., Zhang, S., Wang, H.: Predicting the direction of stock markets using optimized neural networks with Google Trends. Neurocomputing **285**, 188–195 (2018). https://doi.org/10.1016/j.neucom.2018.01.038
27. Qiu, M., Song, Y., Akagi, F.: Application of artificial neural network for the prediction of stock market returns: The case of the Japanese stock market. Chaos, Solitons Fractals **85**, 1–7 (2016). https://doi.org/10.1016/j.chaos.2016.01.004
28. Hafiz, F., Broekaert, J., Torre, D.L., Swain, A.: Co-evolution of neural architectures and features for stock market forecasting: a multi-objective decision perspective. Decis. Support Syst. **174**, 114015 (2023). https://doi.org/10.1016/J.DSS.2023.114015
29. Dingli, A., Fournier, K.S.: Financial time series forecasting: a deep learning approach. Int. J. Mach. Learn. Comput. **7**(5), 118–122 (2017). https://doi.org/10.18178/ijmlc.2017.7.5.632
30. Jing, N., Wu, Z., Wang, H.: A hybrid model integrating deep learning with investor sentiment analysis for stock price prediction. Expert Syst. Appl. **178**, 115019 (2021). https://doi.org/10.1016/J.ESWA.2021.115019
31. Minh, D.L., Sadeghi-Niaraki, A., Huy, H.D., Min, K., Moon, H.: Deep learning approach for short-term stock trends prediction based on two-stream gated recurrent unit network. IEEE Access **6**, 55392–55404 (2018). https://doi.org/10.1109/ACCESS.2018.2868970
32. Nelson, D.M.Q., Pereira, A.C.M., De Oliveira, R.A.: Stock market's price movement prediction with lstm neural networks. In: Proceedings of the International Joint Conference on Neural Networks, vol. 2017, pp. 1419–1426. Institute of Electrical and Electronics Engineers Inc., New York (2017). https://doi.org/10.1109/IJCNN.2017.7966019
33. Carta, S.M., Consoli, S., Piras, L., Podda, A.S., Recupero, D.R.: Explainable machine learning exploiting news and domain-specific lexicon for stock market forecasting. IEEE Access **9**, 30193–30205 (2021). https://doi.org/10.1109/ACCESS.2021.3059960
34. Yuan, X., Yuan, J., Jiang, T., Ain, Q.U.: Integrated long-term stock selection models based on feature selection and machine learning algorithms for China stock market. IEEE Access **8**, 22672–22685 (2020). https://doi.org/10.1109/ACCESS.2020.2969293
35. Jiang, M., Liu, J., Zhang, L., Liu, C.: An improved stacking framework for stock index prediction by leveraging tree-based ensemble models and deep learning algorithms. Phys. A Stat. Mech. Appl. (2020). https://doi.org/10.1016/j.physa.2019.122272
36. Barak, S., Modarres, M.: Developing an approach to evaluate stocks by forecasting effective features with data mining methods. Expert Syst. Appl. **42**, 1325–1339 (2015). https://doi.org/10.1016/j.eswa.2014.09.026
37. Weng, B., Ahmed, M.A., Megahed, F.M.: Stock market one-day ahead movement prediction using disparate data sources. Expert Syst. Appl. **79**, 153–163 (2017). https://doi.org/10.1016/j.eswa.2017.02.041
38. Zhou, P.-Y., Chan, K.C.C., Ou, C.X.: Corporate communication network and stock price movements: insights from data mining. IEEE Trans. Comput. Soc. Syst. **5**, 391–402 (2018). https://doi.org/10.1109/TCSS.2018.2812703
39. Bhanja, S., Das, A.: A Black Swan event-based hybrid model for Indian stock markets trends prediction. Innov. Syst. Softw. Eng. (2022). https://doi.org/10.1007/s11334-021-00428-0
40. Patel, J., Shah, S., Thakkar, P., Kotecha, K.: Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques. Expert Syst. Appl. **42**(1), 259–268 (2015). https://doi.org/10.1016/j.eswa.2014.07.040
41. Wang, W.J., Tang, Y., Xiong, J., Zhang, Y.C.: Stock market index prediction based on reservoir computing models. Expert Syst. Appl. **178**, 115022 (2021). https://doi.org/10.1016/J.ESWA.2021.115022
42. Liu, H., Long, Z.: An improved deep learning model for predicting stock market price time series. Digital Signal Process. Rev. J. (2020). https://doi.org/10.1016/j.dsp.2020.102741
43. Ulrich, J.: TTR: Technical Trading Rules. (2020). R Package Version 0.24.2. https://CRAN.R-project.org/package=TTR
44. Mildenberger, C.D.: What (if anything) is wrong with high-frequency trading? J. Bus. Ethics **186**, 369–383 (2023)
45. Roostaee, M.R., Ali Abin, A.: Forecasting financial signal for automated trading: an interpretable approach. Expert Syst. Appl. **211**, 118570 (2023)
46. Mengshetti, O., Gupta, K., Zade, N., Kotecha, K., Mutha, S., Joshi, G.: Synergizing quantitative finance models and market microstructure analysis for enhanced algorithmic trading strategies. J. Open Innov. Technol. Mark. Complex. **10**(3), 100334 (2024). https://doi.org/10.1016/j.joitmc.2024.100334
47. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
48. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: TensorFlow: large-scale machine learning on heterogeneous systems (2015). https://doi.org/10.48550/arXiv.1603.04467
49. David, S.A., Incio, C.M.C., Quintino, D.D., Machado, J.A.T.: Measuring the Brazilian ethanol and gasoline market efficiency using DFA-Hurst and fractal dimension. Energy Econ. **85**, 104614 (2020). https://doi.org/10.1016/j.eneco.2019.104614
50. Lones, M.A.: How to avoid machine learning pitfalls: a guide for academic researchers. Patterns (5) (2024)

51. James, G., Witten, D., Hastie, T., Tibshirani, R.: An Introduction to Statistical Learning with Applications in R, vol. 112. Springer, New York, NY (2013). https://doi.org/10.1007/978-1-4614-7138-7
52. Chang, C.-C., Lin, C.-J.: LIBSVM: a library for support vector machines. ACM Trans. Intell. Syst. Technol. **2**(3), 1–27 (2011). https://doi.org/10.1145/1961189.1961199
53. Al-Selwi, S.M., Hassan, M.F., Abdulkadir, S.J., Muneer, A., Sumiea, E.H., Alqushaibi, A., Gamal Ragab, M.: Rnn-lstm: from applications to modeling techniques and beyond systematic review. J. King Saud Univ. Comput. Inf. Sci. **36**(5), 102068 (2024)
54. Das, P.: Econometrics in Theory and Practice: Analysis of Cross Section, Time Series and Panel Data with Stata 15.1. Springer, Singapore (2019). https://doi.org/10.1007/978-981-32-9019-8
55. Puneeth, K., Rudagi, S., Namratha, M., Patil, R., Wadi, R.: Comparative study: stock prediction using fundamental and technical analysis. In: 2021 IEEE International Conference on Mobile Networks and Wireless Communications (ICMNWC), pp. 1–4 (2021). https://doi.org/10.1109/ICMNWC52512.2021.9688449
56. Bi, J.: Stock market prediction based on financial news text mining and investor sentiment recognition. Math. Probl. Eng. **2427389**, 1–9 (2022). https://doi.org/10.1155/2022/2427389
57. Alzoubi, M.: Stock market performance: reaction to interest rates and inflation rates. Banks Bank Syst **7**, 189–198 (2022)
58. Larbi Asiedu, E., Mireku-Gyimah, D., Kamasa, K., Otoo, H.: Interest rate, inflation and stock market performance in Ghana: a sector based vector error correction model perspective. Afr. J. Bus. Econ. Res. **16**, 185–206 (2021)
59. Pradhan, R.P.: Development of stock market and economic growth: the G-20 evidence. Eurasian Econ. Rev. **8**(2), 161–181 (2018)
60. Zhang, Z.: Stock Returns and Inflation Redux: An Explanation from Monetary Policy in Advanced and Emerging Markets. Working paper 219, International Monetary Fund (2021)
61. Juodis, A., Karabiyik, H., Westerlund, J.: On the robustness of the pooled CCE estimator. J. Econ. **220**(2), 325–348 (2021). https://doi.org/10.1016/j.jeconom.2020.06.002
62. Juodis, A., Sarafidis, V.: An incidental parameters free inference approach for panels with common shocks. J. Econ. **229**(1), 19–54 (2022)
63. Dhaene, G., Jochmans, K.: Split-panel jackknife estimation of fixed-effect models. Rev. Econ. Stud. **82**(3), 991–1030 (2015). https://doi.org/10.1093/restud/rdv007
64. Granger, C.W.J.: Investigating causal relations by econometric models and cross-spectral methods. Econometrica **37**, 424–438 (2015). https://doi.org/10.2307/1912791
65. Granger, C.W.J.: Some recent development in a concept of causality. J. Econ. **39**, 199–211 (1988). https://doi.org/10.1016/0304-4076(88)90045-0
66. Baum, C.F., Hurn, S., Otero, J.: Testing for time-varying granger causality. Stand Genomic Sci. **22**(2), 355–378 (2022). https://doi.org/10.1177/1536867X221106403
67. Xiao, J., Juodis, A., Karavias, Y., Sarafidis, V., Ditzen, J.: Improved tests for Granger non-causality in panel data. Stata J. (2023). https://doi.org/10.1177/1536867X231162034
68. Dendramis, Y., Kapetanios, G., Tzavalis, E.: Shifts in volatility driven by large stock market shocks. J. Econ. Dyn. Control **55**, 130–147 (2015)
69. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
70. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org (2015). https://www.tensorflow.org/