Multimedia Tools and Applications
https://doi.org/10.1007/s11042-024-20321-9

Forecasting stock market returns using deep learning and time series techniques: a comparative and empirical study using technical indicators

Kalloubi Fahd[1,3] · Hirchoua Badr[2,3] · Labiad Salah Eddine[3] · Aterhi Mouad[3]

Received: 15 January 2023 / Revised: 16 September 2023 / Accepted: 20 September 2024

Abstract

Deep learning models enable learning strategic and accurate behaviors. Nevertheless, training accurate models in dynamic environments, such as stock markets, has been a long-standing challenge. They suffer from partial observability, long-term dependencies, and require merging different handcrafted features. Moreover, selecting the model architecture and fine-tuning different parameters is very challenging. In this paper, we conduct a comparative and empirical study of different possible scenarios. We demonstrate and provide a solid recommendation for possible network architectures. Precisely, we compare different recurrent neural network models alongside time series techniques and report their influence on out-of-sample stock returns. Furthermore, we run an empirical study to determine the best features and hyperparameters using several feature importance techniques. Strong technical indicators have fueled the input prices to push the input representation forward. Extensive experiments have been presented to evaluate the performance of different models, and the results are critically analyzed and empirically verified. The results demonstrate that bidirectional recurrent neural networks and the autoregressive integrated moving average with explanatory variable model provide better performances in predicting the following day's closing price and provide better references and insights for future investments.

Keywords Machine learning · Deep learning · Recurrent neural networks · Time series forecasting · Predictive modeling · Stock price forecasting

✉ Hirchoua Badr
badr.hirchoua@gmail.com

Kalloubi Fahd
kalloubi.f@ucd.ac.ma

Labiad Salah Eddine
salaheddine.labiad@um6p.ma

Aterhi Mouad
mouad.aterhi@um6p.ma

1 LTI Laboratory, National School of Applied Sciences, El Jadida, Morocco

2 National Higher School of Arts and Crafts (ENSAM), Hassan II University (UH2), Casablanca, Morocco

3 Modeling, Simulation and Data Analysis, Mohammed VI Polytechnic University, Ben Guerir, Morocco

1 Introduction

Stock markets collect information from a complex and ever-changing real-world environment. Automated trading is a well-established process that involves executing multiple orders into an exchange [1]. The trading happens in an ongoing double auction containing buyers and sellers continuously interacting with an open order book on an exchange. The general rule is to buy at low prices and sell at high ones. The stock exchanges are associated with non-linearity, highly fluctuating factors, and a constant change of market details. The traditional way of investing in the stock market was about understanding the company from the core and putting a value on future cash flows. This way, investors can only invest in a few companies, at max, a few dozen. However, while technology changes, it sheds light on different angles, including automated and dynamic investing. It combines various fields of solving this issue, involving statistics, computer science, and mathematics. With the great success of machine learning (ML) / deep learning (DL) in many areas, ML-based financial and stock forecasting has gained more attention in recent years and has achieved tremendous success [2–4].

In stock and financial markets environments, the higher profits come with higher risks involved with stock exchange rates [5]. Traders need to predict trends in stock market behavior for a correct decision to either sell or hold the stock they possess or buy other stocks instead. These three possible execution scenarios -sell, buy, and hold- are the actions to output in a Deep Reinforcement learning (DRL) policy network. The policy network takes the state encoding as input and outputs the best action directly. If we can predict the price trend, we can efficiently execute the correct action [6]. If stock traders correctly predict stock price trends, they can realize significant profits [7]. A hallmark of an intelligent trading system is its capacity to rapidly adjust to novel and unusual situations and predict the following price trends in complex and uncertain situations [1]. Therefore, automated trading systems must meet some standards to master the prediction task in dynamic environments, such as stock markets, and make profitable orders. On the one hand, it must have an inherent adaptive and adjustable nature. If the price position changes, the system can easily anticipate and adapt, thus improving the trading strategy. On the other hand, it should respect the allocation made by a given investor; therefore, some works combine the financial assets price prediction and the portfolio allocation in one unified process to produce fully autonomous systems [8].

The DL-based trading system is formulated as follows: at each time step, the agent, the neural network, receives the current price position as an input and predicts as output the next price. Given this, the neural network components, such as the layers types, their size, the batch size, and different hyperparameter values, must be tuned and incorporated to work as a single unit to provide accurate and precise predictions. Moreover, at each time step, the input size must cover as many influential prices as possible and more minor to cover only sufficient changes. Next, the input details are not fully observable, but instead, they are partially observable. Thus, dynamic decision-making is more challenging due to the need for more state details, as the agent needs an intrinsic curiosity that can enhance the input and drive the learning process to make correct real-time decisions simultaneously. Consequently, the DL-based trading system can directly predict the next price or the following best action. However, for both scenarios, we need the same feature extraction process. The only difference is whether the last layer predicts the action (i.e., a classification problem) or the next price (i.e., a regression problem).

Stock and financial trading models can be split into two leading families: price prediction [9, 10], and stock-trading point prediction [11]. Most works in the first family focus only on forecasting the following day's prices, whereas some works use that prediction as a signal for a directional trading strategy [9, 10, 12, 13]. On the other hand, others' work exploits time series forecasting and regression models to predict the trend of the price movement [14, 15]. The second family focuses on generating the action signal: sell, hold, or buy. In this paper, we adopt the first form by developing multiple regression-based models for forecasting stock price time series.

In addressing the aforementioned problems, this paper analyzes the DL and classical time series forecasting models in-depth. Beforehand, we investigate the possible values for different neural network parameters and hyperparameters. We have considered four architectures: LSTM, GRU, BiLSTM, and BiGRU. In addition, we have considered traditional time series techniques to highlight the main difference between the two technologies. Next, we investigated the input size, also known as the window size. This parameter has a significant influence on the prediction accuracy. We have decided to include extra past price as a unique attention mechanism, proving its inexorability in such a configuration. It gives the agent an additional signal to focus more on the area of change. After conducting extensive experiments using multiple time series and machine learning models, we have noted a need for extra features to fuel and enhance the input. In other words, the agent needs a curiosity mechanism to drive the learning process. Therefore, we have included technical indicators directly related to stock market changes. Equally important, the obtained results are compared to find the most consistent predictors and, eventually, the most appropriate model. The experiment results are conducted on Apple and Google datasets returns between March 2015 and March 2021. These datasets have been extended with some technical indicators, which have been shown to give a solid predictive capacity [16]. The performance results are presented in terms of Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and Root Mean Square Error (RMSE). For reproducibility purposes, we made the code of this paper available in a GitHub repository[1].

To the best of our knowledge, this is the first comparative study using extended DL (i.e., BiLSTM and BiGRU) and time series prediction models by conducting an in-depth empirical analysis. The contributions of this paper are summarized as follows:

- We conducted a comparative study on different RNN models: LSTM, BiLSTM, GRU, and BiGRU models and the ARIMAX model on financial time series forecasting.
- We investigated the window size as a new attention mechanism, which is tuned to cover as many influential prices as possible and more minor to cover only sufficient changes.
- We analyzed the technical indicators as an intrinsic curiosity mechanism that fuels and enhances the inputs.
- We leveraged the time series of stock price data, given a combination of market data (Open, High, Low, Close, and Volume) and several technical indicators, to predict the stock price.
- We conducted an extensive empirical analysis to choose the best set of hyperparameters and also the best combination of features.

The remainder of this paper is organized as follows. Related work in stock price returns prediction is presented and discussed in Section 2. Section 3 explains our methodology and describes each step in detail. We provide the experimental results and discussion in Section 4. Finally, Section 5 concludes this paper by providing suggestions for future work.

[1] https://github.com/s0v1x/AriStocks

2 Related works

Many research works are conducted for stock market prediction [17, 18]. Among the proposed techniques, ML methods have been widely used in forecasting time series because they are data-driven, self-adaptive methods that can capture time series' nonlinear behaviors without any statistical assumptions about the data [19]. Traditional statistical methods assume that time series are generated from a linear process. However, real-life financial and stock time series data are more complex, nonlinear, and non-parametric [20].

Support vector machine (SVM) is one of the most used supervised machine learning models [21]. Henrique et al. [22] have used SVR to predict the stock prices for a different market. Experiment results show that SVR has significant predictive power. Nayak et al. [23] have proposed a framework by incorporating SVM with K-nearest neighbor (KNN). The proposed approach has been used to predict the Indian stock exchange market. SVM has been utilized to predict future profit or loss and estimate the stock value for one day, week, and month. The model has performed well for high-dimensional feature vectors and handled the classification methods' errors and performance. The SVM-KNN model has outperformed the mentioned models by removing the need to tune multiple parameters. Also, Tay et al. [19] have proposed a two-stage neural network (NN) architecture constructed by combining SVMs with a self-organizing feature map for financial time series forecasting, demonstrating that the proposed method achieves both significantly higher prediction performance and faster convergence speed in comparison with a single SVM model. Many ML methods can capture complex nonlinear relationships among relevant factors without prior knowledge about the input data. Thakur and Kumar [15] have proposed a hybrid system for automatic trading that integrates weighted multicategory generalized eigenvalue SVM and RF algorithms. Chen et al. [24] have used stock price data from the Chinese stock market to compare the price prediction of traditional NN with DL. They have found that DL's prediction performance was better than traditional NNs. Jayanthbalaji et al. [25] have used DL models for stock price forecasting. It uses fourteen different types of learning models, which use different learning techniques like long short-term memory (LSTM), extreme learning machines (ELMs), gated recurring units (GRU), and convolution neural networks (CNN). It is observed that GRU-based models give better directional accuracy than other models. Moreover, Sezer et al. [26] have found that RNN-based models are the most used DL models for stock price forecasting.

Baek and Kim [10] have proposed a modular architecture using two LSTM networks that beat the buy-and-hold strategy. Bao et al. [9] have proposed a three stages novel DL framework which fused the wavelet transforms (WTs), stacked autoencoders (SAEs), and long-short-term memory (LSTM) for stock price forecasting. Their method consists of three stages. First, the framework utilize the WTs to eliminate noise in time series data. Next, they apply SAEs to generate deep high-level features to predict the stock price. Last, they fit high-level denoising features to LSTM to forecast the next day's closing price. Zhou et al. [12] have introduced a two stages end-to-end approach. It consists of the empirical mode decomposition and a factorization-machine-based neural network (NN) to predict the price trend. This approach has been compared to an NN, a factorization-machine-based NN, an empirical mode decomposition-based NN (EMD2NN), and a wavelet de-noising-based backpropagation NN model. The results reveal that the EMD2NN model outperformed the other ones.

Krauss et al. [13] have investigated the usefulness and validity of deep NNs, random forest (RF), gradient-boosted trees, and other ensembles of these models in the statistical arbitrage context. Their model records daily one-day-ahead trading signals based on the probability forecast of stock and outperforms the general market. The random forests surpass gradient-boosted trees and deep neural networks. However, they have declared that cautious hyperparameter optimization may produce promising outcomes for tuning-intensive DL models. Zhou et al. [14] have introduced a learning architecture for forecasting and trading stock indices (LogR2GBDT). Their model cascades a logistic regression (LogR) model onto a gradient-boosted decision trees (GBDT) model. The model outperforms the benchmark models, and more impressively, it yields statistically and economically significant improvements in terms of exploiting simple trading strategies. Chalvatzis and Hristu-Varsakelis [27] have tuned their proposed prediction model to enhance profitability rather than accuracy. The results mark that their best overall model achieves very high cumulative returns and outperforms the benchmark buy-and-hold strategy and other recent works. Fischer and Krauss [28] have investigated the daily directions of S&P and compared different ML models. They have recorded mean daily returns of 0.46 and a Sharpe ratio of 5.8, which have pointed out the low risk/high reward prior to transaction costs.

Despite the abovementioned studies, the deep learning-based trading agent remains an area for improvement. On the one hand, many works have considered the SVM model [15, 19, 23]. However, this model cannot capture the highly non-linearity of the stock market data. It is computationally expensive and easily prone to overfitting when dealing with high-dimensional data. SVMs are not designed to address the dynamic nature of stock market data and are sensitive to imbalanced data. On the other hand, Krauss et al. [13] and Zhou et al. [12] have yet to consider RNN models, which have proven their performance in the modelization of time series data. Even though Baek and Kim [10] have used the LSTM model, they have considered the WTs to eliminate noise in time series data, which also removes solid and predictive signals. Chalvatzis and Hristu-Varsakelis [27] have considered the LSTM model and omitted the other strong models, as we have demonstrated in this work. Considering these limitations, we have conducted an in-depth study to push the automated trading-based DL model state-of-the-art forward. Precisely, we have investigated different RNN models alongside traditional time series techniques with different hyperparameters values. Moreover, we considered two additional mechanisms to enhance the input representation: the attention mechanism and curiosity-driven learning.

The following section is dedicated to the methodology for analyzing and forecasting stock price time series.

3 Methodology

This section presents our methodology for analyzing and forecasting stock price time series. The main drawback of most existing methods is the lack of other essential variables that directly or indirectly affect the stock price. Stock markets contain an unknown number of active agents, each with a specific behavior. Therefore, the following price prediction in such an environment is challenging. Furthermore, in some cases, the prediction is impossible due to the partial observability of market data. Our approach pushes the prediction process forward by fueling the input state with technical indicators. These technical indicators include Trend, Volatility, Momentum, Volume, and Other indicators, detailed in Section 3.2. Moreover, we track several technical indicators to highlight the company's performance in the stock market. These technical indicators are used to expand our stock price data and improve the performance of our prediction models [16].

3.1 Deep learning and time series models

3.1.1 Long-short term memory

Long-Short Term Memory (LSTM) is a network model designed to solve the longstanding problems of the explosion and vanishing of the gradient in recurrent neural networks (RNN) [29]. It has been widely used in speech recognition and text analysis. It has its memory type and can make relatively accurate forecasting [30]. Recently, LSTM has also been widely adopted in stock market prediction [31].

A standard RNN with a simple internal structure has only one repeating module. However, the LSTM modules are similar to the standard RNN modules and operate in a particular interactive manner. The LSTM memory cell consists of three gates: the forget gate, the input gate, and the output gate. Figure 1 highlights the LSTM architecture with different links between parts.

The LSTM calculation process can be divided into four parts. First, the **forget gate** inputs the output of the previous moment h_{t-1} and the current moment X_t input value. The output value of the forget gate is obtained after the intervening of the sigmoid function. Specifically,

$$f_t = \sigma(W_f.[h_{t-1}, X_t] + b_f), \tag{1}$$

where f_t ranges between 0 and 1, W_f and b_f are, respectively, the weights and the bias associated with the forget gate.

Next, the **input gate** takes the input value of the current moment X_t and the output of the previous moment h_{t-1}. The output value of the input gate is received after applying the sigmoid function as follows:

$$i_t = \sigma(W_i.[h_{t-1}, X_t] + b_i), \tag{2}$$

where i_t ranges between 0 and 1, W_i and b_i are, respectively, the weights and the bias associated with the input gate.

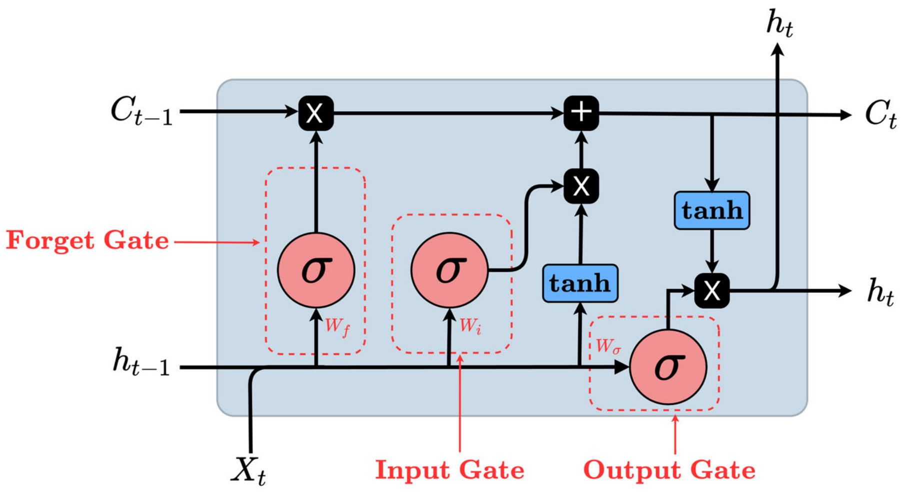

Fig. 1 Long-Short Term Memory (LSTM) memory cell

The **update of the current cell state**, the third component, takes the same input as the two previous gates as inputs to the hyperbolic function **tanh**, which returns the current candidate state, for which the current cell state is updated after calculation. Specifically,

$$C_t = f_t * C_{t-1} + i_t * tanh(W_c[h_{t-1}, X_t] + b_c)), \tag{3}$$

where the value of C_t ranges between 0 and 1, f_t is the output value of the forget gate, C_{t-1} is the cell state of the previous moment, i_t is the output value of the input gate, W_c and b_c are, respectively, the weights and the bias associated with the current candidate state.

Last, the **output gate** returns the calculations as follows:

$$o_t = \sigma(W_o.[h_{t-1}, X_t] + b_o), \tag{4}$$

where the value of o_t ranges between 0 and 1, W_o and b_o are, respectively, the weights and the bias associated with the output gate, X_t is the input value of the current moment, and h_{t-1} is the output value of the previous moment.

LSTM returns as an output value, h_t, the combination of the previous results, as presented in the following formula:

$$h_t = o_t.tanh(C_t), \tag{5}$$

where o_t is the output gate's value, and C_t is the current state of the cell.

3.1.2 Gated recurrent unit

Gated Recurrent Unit (GRU) is a recurrent neural network based on LSTM, proposed by [32]. It re-transforms the LSTM architecture by combining the forget gate and the input gate into a single gate called the update gate. Furthermore, they combined the hidden and cell states in LSTM in a single gate called the reset gate. Figure 2 presents the GRU memory cell.

The GRU neural network is a sequential neural network that determines the output of the current moment based on the input value at the same moment and the output value of the previous moment. A GRU neural network has two control gates, reset and update gates.

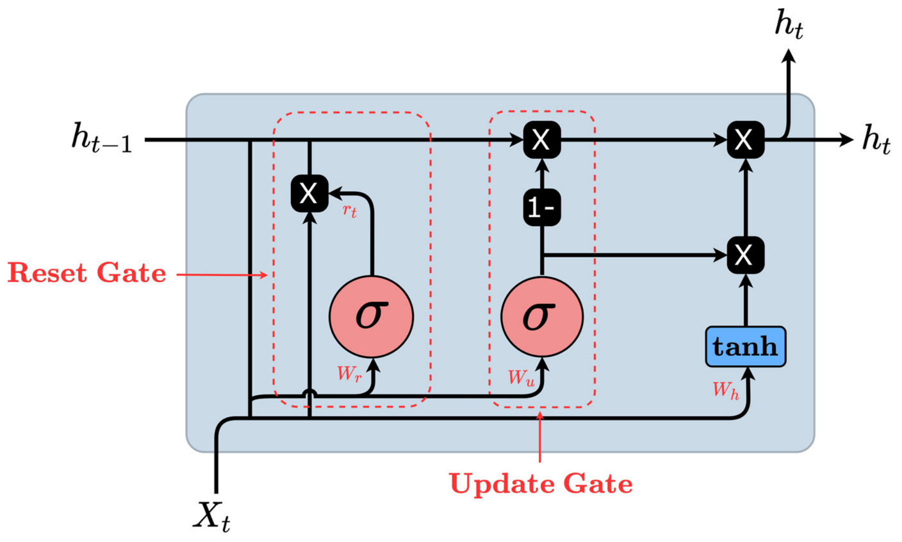

Fig. 2 Gated Recurrent Unit (GRU) memory cell

The reset gate controls how much information is passed that the network must forget. The previous hidden state h_{t-1}, concatenated with the input data X_t, goes through the sigmoid function as shown in the following formula:

$$r_t = \sigma(W_r.[h_{t-1}, X_t] + b_r), \tag{6}$$

where r_t ranges between 0 and 1, W_r and b_r are, respectively, the weights and the bias associated with the reset gate. On the other hand, the update gate acts the same as the LSTM's forget and entry gate. It decides what information to keep and what to forget. The input data X_t and the previous hidden state h_{t-1}, are concatenated and passed through the sigmoid function, as shown in the following formula:

$$u_t = \sigma(W_u.[h_{t-1}, X_t] + b_u), \tag{7}$$

where u_t ranges between 0 and 1, W_u and b_u are, respectively, the weights and the bias associated with the update gate. As a result, GRU returns as an output value h_t, a combination of the previous results. It is presented in the following formula:

$$h_t = (1 - u_t) * h_{t-1} + u_t * tanh(W_h.[r_t * h_{t-1}, X_t]), \tag{8}$$

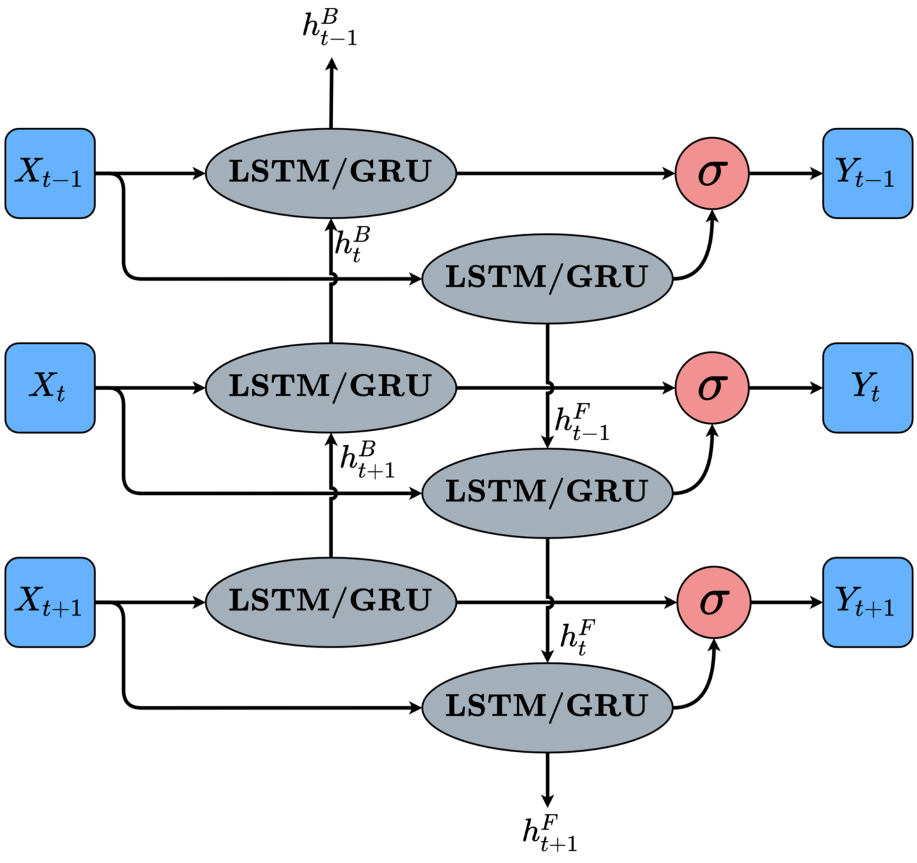

Fig. 3 BiLSTM and BiGRU architectures

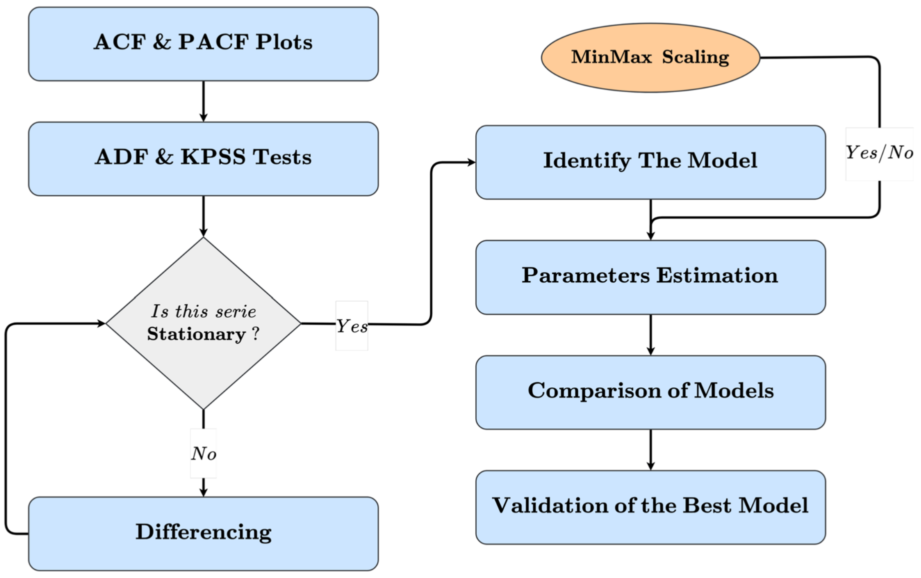

Fig. 4 Procedure of applying ARIMA model

where u_t and r_t are, respectively, the output values of the update and the reset gates, W_h are the weights associated with the hidden state, and $*$ is denoted as the Hadamard [2] product.

3.1.3 Bidirectional LSTM and bidirectional GRU

Bidirectional LSTM (respectively Bidirectional GRU) is a sequence processing model consisting of two LSTMs (respectively GRUs): one takes the input forward and the other backward.

BiLSTMs and BiGRUs effectively increase the amount of information available to the network, improving the content available to the algorithm [33]. Figure 3 illustrates the architecture for the two models.

3.1.4 Autoregressive integrated moving average with explanatory variable

The Autoregressive Integrated Moving Average (ARIMA) model [34] is a fundamental univariate time series model. The ARIMA model is made up of three key components. To begin with, the autoregressive component (AR) represents the relationship between the current dependent variable at lagged periods. Nest, the integrated component (I) refers to transforming the data by subtracting past values of a variable from the current values to make the data stationary. Last, the moving average component (MA) refers to the dependency between a stochastic term's dependent variable and past values.

The ARIMA does not support seasonal data; it expects either not seasonal data or has the seasonal component removed, e.g., seasonally adjusted via methods such as seasonal

[2] Jacques Salomon Hadamard 8 December 1865, 17 October 1963, was a French mathematician who made significant contributions in number theory, complex analysis, differential geometry, and partial differential equations.

Table 1 Partial sample of Apple's dataset, where CR stands for Cumulative Returns

Date	Open	High	...	EMA50	SMA5	...	ATR50	CR
2015-03-17	31.475000	31.830000	...	30.396444	31.113499	...	0.699026	16.198665
...	...	...	...	...	...	...	...	...
2015-03-23	31.780001	31.962500	...	30.605605	31.806000	...	0.692399	16.354157
...	...	...	...	...	...	...	...	...
2017-09-12	40.652500	40.990002	...	39.138425	40.208000	...	0.608830	47.132535
...	...	...	...	...	...	...	...	...
2021-03-29	121.650002	122.580002	...	125.239687	121.163998	...	3.600508	344.123293
2021-03-30	120.110001	120.400002	...	125.030288	120.635998	...	3.579098	338.671918

differencing. For this reason, Seasonal Autoregressive Integrated Moving Average (SARIMA or Seasonal ARIMA) has been proposed. They explicitly extend ARIMA to support univariate time series data with a seasonal component. Figure 4 presents the application process of the ARIMA model.

Although ARIMA and SARIMA are compelling models for forecasting univariate time series, the multivariate time series is a critical aspect to lean towards. Therefore, instead of implementing ARIMA or SARIMA models, we opted for ARIMAX. It is an extension of the ARIMA model that involves adding exogenous components, or variables, where the values are imposed on the model and affect the target variable. Pankratz et al. [35] refer to the ARIMAX model as dynamic regression.

To find the best ARIMAX model, Auto-ARIMA is used to search for the possible models by optimizing for a given information criterion (i.e., Akaike Information Criterion, Corrected Akaike Information Criterion, or Bayesian Information Criterion) and returns the best model which minimizes the value.

3.2 Data description

To test the developed architecture, we consider two datasets-Apple's and Google's stock price time series. Both datasets include 1521 trading days from March 17th, 2015, to March 30th, 2021. Tables 1 and 2 highlight a partial sample of both datasets. For better visibility, Fig. 5 illustrates both datasets over the considered periods. The first 1216 trading days are used to train the model. The rest of the datasets is used to test the model.

Table 2 Partial sample of Google's dataset

Date	Open	High	...	EMA50	SMA5	...	ATR50	CR
2015-03-17	27.914499	28.032000	...	27.296855	26.877100	...	0.534381	5.298838
...	...	...	...	...	...	...	...	...
2015-03-23	28.299999	28.482000	...	27.438042	28.177600	...	0.519731	6.764225
...	...	...	...	...	...	...	...	...
2017-09-12	47.346001	47.404499	...	47.282310	47.232600	...	0.674860	78.764982
...	...	...	...	...	...	...	...	...
2021-03-29	101.108002	102.460503	...	99.641641	101.768400	...	2.489410	286.326100
2021-03-30	102.649002	103.133499	...	99.746812	101.819801	...	2.466142	286.454529

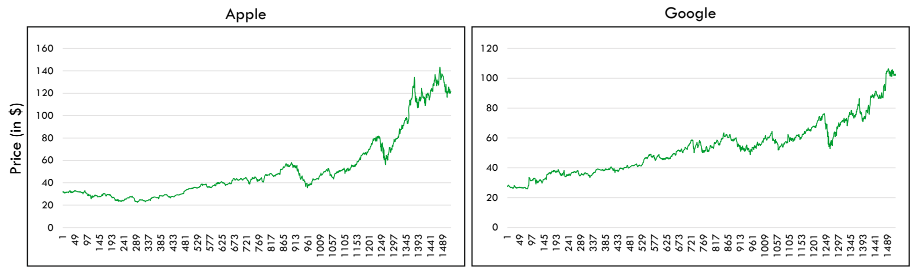

Fig. 5 Apple and Google prices from March 17th, 2015, to March 30th, 2021

This dataset provides numerous features influencing the close price. Our approach includes adding technical indicators to extract the maximum financial information possible regarding the usual features, such as the open, high, low, adjusted close, and volume. To do so, we define the following types of technical indicators:

- Trend Indicators:
  - Simple Moving Average (SMA) calculates the average of a selected range n of closing prices by that range of values. Specifically,
  $$SMA_n^t = \frac{1}{n}\sum_{t=1}^{n} y^t, \tag{9}$$
  where y^t is the closing price at time t.
  - Exponential Moving Average (EMA) is a moving average that places a greater weight and significance on the most recent data points, as follows:
  $$EMA_n^t = \alpha y^t + (1-\alpha)EMA_n^{t-1}, \tag{10}$$
  where $\alpha = \frac{2}{n+1}$, y^t is the closing price at the current time, n is the number of days, and EMA_n^{t-1} is the EMA value at time $t-1$.
  - Average Directional Index (ADX) is a technical indicator used by some traders in the financial market to determine the strength of a trend in the stock.
  - Commodity Channel Index (CCI) is a technical indicator measuring the difference between current and historical average prices.
- Volatility Indicators:
  - Average True Range (ATR) is a technical analysis indicator that measures market volatility by decomposing the entire range of an asset price for that period.
  - ULCER Index (UI) measures the downside of investment risk in terms of the duration for which the price is declining and the price depth.
- Momentum Indicators:
  - Rate Of Change (ROC) describes the percentage of change in value over a defined period of time, and it represents the momentum of a variable, as described in the following formula:
  $$ROC_n^t = \left(\frac{y^t}{y^{n-t}} - 1\right) \times 100, \tag{11}$$

  - Relative Strength Index (RSI) is an indicator used in technical analysis that measures the magnitude of recent price changes to evaluate conditions of overbought or oversold in the price of a stock.

- Volume Indicators:
  - Force Index is an indicator that relies on the amount of power influencing the stock price. Specifically,
$$FI_n^t = (y^t - y^{t-1}) \times V_n, \tag{12}$$
  - Money Flow Index (MFI) is a technical indicator that generates signals indicating the overbought or oversold of a stock.
  - On Balance Volume (OBV) is an indicator that shows the investors' sentiment based on the volume related to the stock, which can indicate the investors' opinion of the stock, influencing the stock price outcome.

- Other indicators:
  - Accumulation/Distribution (A/D) indicator provides insight into the strength of a trend. It uses the asset's price and volume to determine whether the stock is accumulated or distributed.

3.3 Models training and forecasting process

To train the model, we have considered the following process:

1. Input data: Create and re-assemble the datasets of historical prices. Then, we calculate the adopted technical indicators.
2. Data reconstruction (for RNNs): The RNN models receive the input data in a particular format. It requires the (S, W, F) format, where S is the number of samples, W is the window size, and F is the number of features. Note that the window size parameter affects the accuracy of the models; therefore, it requires cautious tuning.

Table 3 Important set of features

Method	Technique	Features Selected
Filter	Pearson's correlation	open, high, low, adj close, cumulative returns, SMA and EMA (for 5, 10, 15, 20, 25, 30, 35, 40, 45, and 50 days), ATR (for 25, 30, 35,40, 45, and 50 days), long-term debt to capital, debt to equity ratio, return on equity
	F-regression test	open, high, low, adj close, cumulative returns, SMA (for 5, 10, 15, and 20 days), and EMA (for 5, 10, 15, 20, 25, and 30 days)
Embedded	Random Forest Regressor	open, high, low, adj close, EMA5, A/D index, and cumulative returns
	Lasso regularization	EMA (for 10, 15, 20, and 50 days), and A/D index
Wrapper	Bidirectional Elimination	cumulative returns, low, open, and high

3. Data normalization: Considering the gap in data values, we need to normalize the data to train different models stably. The data normalization adopted for this approach is shown in the following formula:
$$x'_{i,j} = \frac{x_{i,j} - min_j}{max_j - min_j}, \tag{13}$$
where $x_{i,j}$ is the actual price, $x'_{i,j}$ is the normalized price, and min_j and max_j are the minimal and the maximal values of the j^{th} feature, respectively.
4. Feature importance: In this paper, we have proposed five sets of features resulting from feature selection methods: filter, embedded, and wrapper. Each set contains the essential features, as shown in Table 3.
These features are denoted, from top to bottom, as F1, F2, F3, F4, and F5. This notation is adopted for the following sections.
5. Initializing the models: We have adopted the same architecture for the DL models (LSTM, GRU, BiLSTM, and BiGRU). The models are trained to find the best combination simultaneously. Figure 6(a) highlights the architecture without using the dropout layers. Next, the dropout layers are implemented as shown in Figure 6(b).
The architecture is sequential of 2 hidden layers. The first is initialized with 50 units, whereas the other is 100 units. Furthermore, a rate of 20% for dropout is considered after each hidden layer.
6. Hyperparameter tuning: A hyperparameter tuning step is required to get the best results. Precisely, we tune the batch size, the number of epochs, and the window size.
As for the ARIMAX model, the Auto-ARIMA approach automatically finds the optimal parameters.
7. Models' evaluation and comparison: The DL and classical time series models mentioned above are compared based on the results of the evaluation phase. The models are compared after choosing the best parameters for each model. In addition, we took the best set of features to evaluate each model.

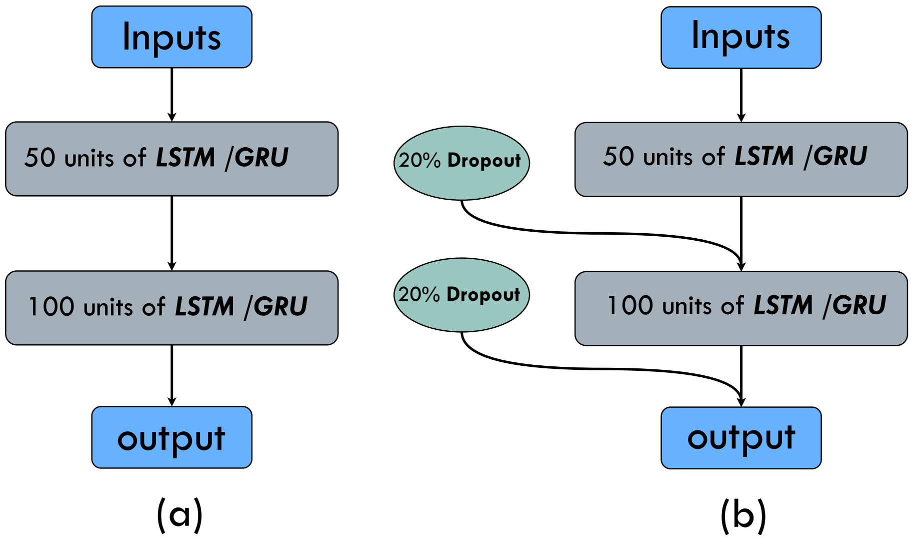

Fig. 6 (a) RNNs models' architecture without drop out. (b) RNNs models' architecture with drop out

Table 4 ARIMAX model performance on Apple data

		F1	F2	F3	F4	F5
Without Normalization	RMSE	37.48	20.89	28.01	15.74	**3.065**
	MAE	30.98	17.39	23.33	13.19	**2.324**
	MAPE	28.55	15.83	20.84	12.46	**2.301**
With Normalization	RMSE	37.48	18.34	28.01	19.49	27.57
	MAE	30.98	15.27	23.33	16.4	22.98
	MAPE	28.55	14.09	20.84	15.2	20.57

4 Results and discussion

To assess the effectiveness of our approach, we executed the LSTM, GRU, BiLSTM, BiGRU, and ARIMAX models on the same training set and test set. We have used the mean squared error loss function optimized by the Adam optimizer for all experiments. All experiences were conducted with one single-core hyper-threaded Xeon processor @2.3Ghz and 12 GB of memory. We have tested and observed different values for splitting the training and testing datasets. The best-split percentage is to use 80% for training different models and the rest 20% for testing.

4.1 Autoregressive integrated moving average with explanatory variable

The evaluation procedure of the ARIMAX model consists of finding the optimal parameters (p, d, and q) that record the lowest error metrics. Note that the evaluation is derived using a different set of features. The ARIMAX model is sensitive to data transformation, which affects the model's performance. The ARIMAX model's performance results are shown in Tables 4 and 5.

Interestingly, the ARIMAX model has performed much better by combining no-order Auto-Regressive (p=0), one day of differencing (d=1), and a fourth-order moving average (q=4) model. Moreover, the best results are given without transforming the data and by using the F5 set of features which targets the cumulative returns and the OHL (Open, High, and Low) features.

Table 4 presents the ARIMAX model performance on Apple data. Using the set of parameters above and without data transformation, the Root Mean Square Error (RMSE), Mean Absolute Percentage Error (MAPE), and Mean Absolute Error (MAE) tremendously

Table 5 ARIMAX model performance on Google data

		F1	F2	F3	F4	F5
Without Normalization	RMSE	29.80	13.56	12.10	13.94	**2.862**
	MAE	20.26	10.08	9.12	10.17	**2.136**
	MAPE	22.85	12.37	11.38	12.35	**2.668**
With Normalization	RMSE	33.39	13.46	12.07	21.41	10.18
	MAE	22.48	10.00	9.09	15.95	7.67
	MAPE	25.14	12.27	11.33	18.53	9.52

decreased from 27.57, 20.57, and 22.98 to 3.065, 2.301, and 2.324, respectively. Table 5 shows that the ARIMAX model has performed better without data transformation when predicting Google stock prices. The F5 feature set consistently produced the most accurate results. The RMSE, MAPE, and MAE values remained relatively low across all feature sets. When the data was not transformed, the RMSE, MAPE, and MAE values substantially decreased from 10.18, 9.52, and 7.67 to 2.862, 2.668, and 2.136, respectively.

The predictions based on the ARIMAX model for both datasets are compared to the actual close price as shown in Fig. 7. These results confirm and demonstrate our proposal. The main objective in stock markets, where the prices are changing heavily, is to predict whether the price will go up or down. Based on this assumption, Fig. 7 establishes our hypothesis of predicting the price trend. The F5 features record the lowest error values for both datasets and strengthen the cumulative return flexibility and representativity alongside the pricing data. The normalization mechanism transforms the big and small prices into smaller intervals. Therefore, in the context of the stock market, this behavior hides strong patterns that have a decisive representation of different inputs.

4.2 Recurrent neural network models performance

The RNNs require tuning a significant number of parameters. Therefore, we suggested a step-by-step approach to evaluate such models and find the best parameters that provide the best predictions. In the first step, we create the architectures presented in Fig. 6 for LSTM, GRU, BiLSTM, and BiGRU models. In our approach, we chose an architecture of 50 units in the first hidden layer and 100 in the second. Next, we apply two dropout layers with a 20% rate each. These dropout layers randomly set the input unit rate to 0 at each training step; this helps prevent overfitting. The inputs different from 0 are multiplied by 1.25, so the sum of the inputs remains unchanged. Moreover, a necessary callback for an early stop is required. Therefore, we chose an early stopping after 10 epochs of 1000 non-changing loss function results.

In the second step, we identify the set of features and the batch size that give the lowest error values. For this, and due to the stochastic nature of the DL architectures, considering multiple runs was the key to obtaining the optimal model. The method consists of running the model multiple times and then reporting the average of the results of the evaluation metrics (RMSE, MAE, and MAPE). In our case, we chose an average of 10 values for each metric.

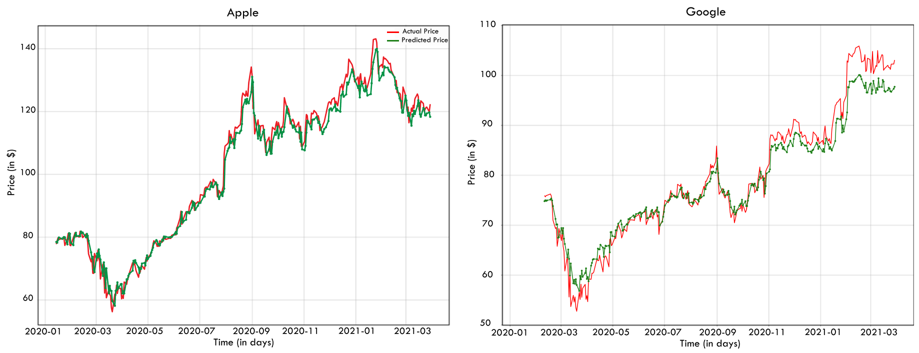

Fig. 7 Comparison of the predicted value and the real value for ARIMAX for Apple and Google datasets

Table 6 Evaluation of LSTM, GRU, BiLSTM, and BiGRU models with different parameters on Apple Data

Batch Size	Architecture	Drop out	Features	RMSE	MAE	MAPE
128	**LSTM**	**F**	**F3**	**3.315**	**2.499**	**2.494**
	GRU	**T**	**F2**	**3.497**	**2.564**	**2.525**
	BiLSTM	**T**	**F3**	**3.664**	**2.762**	**2.743**
	BiGRU	**T**	**F5**	**4.224**	**3.294**	**3.187**
256	LSTM	T	F5	4.415	3.247	3.174
	GRU	F	F4	5.521	4.195	4.134
	BiLSTM	F	F3	4.342	3.24	3.087
	BiGRU	F	F5	6.714	5.298	4.786
512	LSTM	F	F5	4.381	3.226	3.181
	GRU	T	F2	6.862	5.502	4.996
	BiLSTM	T	F4	7.403	5.979	5.781
	BiGRU	F	F5	6.739	5.356	4.877
1024	LSTM	T	F4	6.68	4.939	5.003
	GRU	F	F5	5.855	4.447	4.073
	BiLSTM	F	F5	8.403	6.528	5.819
	BiGRU	T	F5	5.612	4.265	3.974

We evaluated each model with different parameters (batch size, set of features, and dropout) with a fixed window size of 5 days. Table 6 shows the evaluation of all models with different parameters on Apple Data. The results show that the 128-batch size gives the best results among the considered values 128, 256, 512, and 1024. Moreover, the sets of

Table 7 Evaluation of LSTM, GRU, BiLSTM, and BiGRU models with different parameters on Google Data

Batch Size	Architecture	Drop out	Features	RMSE	MAE	MAPE
128	**LSTM**	**F**	**F5**	**3.591**	**2.724**	**3.318**
	GRU	**T**	**F5**	**2.572**	**3.282**	**3.126**
	BiLSTM	**F**	**F5**	**3.477**	**2.65**	**3.186**
	BiGRU	**T**	**F5**	**3.463**	**2.738**	**3.325**
256	LSTM	F	F5	4.359	3.363	4.031
	GRU	F	F5	3.982	3.096	3.681
	BiLSTM	T	F5	4.341	3.398	4.051
	BiGRU	T	F5	3.887	3.032	3.629
512	LSTM	F	F5	4.816	3.775	4.526
	GRU	F	F5	4.319	3.445	4.101
	BiLSTM	T	F5	7.608	6.283	7.405
	BiGRU	F	F5	4.279	3.409	4.074
1024	LSTM	F	F5	7.24	5.87	6.931
	GRU	F	F5	4.027	3.014	3.574
	BiLSTM	F	F5	5.789	4.56	5.42
	BiGRU	T	F5	3.641	2.728	3.273

Table 8 Evaluation of LSTM, GRU, BiLSTM, and BiGRU models for different Time Steps (TS) on Apple Data

TS	LSTM			BiLSTM			GRU			BiGRU		
	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE
5	3.867	3.011	2.973	**3.290**	**2.424**	**2.425**	3.958	2.928	2.834	**3.080**	**2.310**	**2.317**
10	**3.754**	**2.886**	**2.771**	3.291	2.431	2.425	3.987	2.965	2.847	3.539	2.719	2.679
15	4.925	3.844	3.794	3.717	2.824	2.813	3.760	2.778	2.676	3.419	2.610	2.593
20	3.962	3.150	3.083	4.967	3.793	3.704	5.102	4.009	3.662	3.532	2.649	2.617
25	4.058	3.078	2.954	4.627	3.607	3.514	**3.625**	**2.703**	**2.640**	3.259	2.469	2.435
30	4.138	3.141	3.079	5.619	4.400	4.156	4.797	3.668	3.359	3.543	2.700	2.638

features that provide the best results are F2, F3, and F5. Considering the batch size of 128, the error values are remarkably close for the LSTM and BiLSTM models that employ the F3 set of features. The GRU and BiGRU have employed different features, the F2 and F5, for the same batch size value. In addition, the error values are slightly different compared to LSTM and BiLSTM. Table 7 highlights the evaluation of all models with different parameters on Google Data. The results confirm that the batch size of 128 provides the best results in terms of error values. Interestingly, as stated in the ARIMAX evaluation, the Google dataset reveals that the best set of features with all batch size values is the F5 features. The Google actual prices in the considered period are compact (prices are not changing heavily) with less volatility, which provides the necessary signal to the model to make good predictions, meaning lower error values.

The window size or time step should be extended to cover as many prices as possible and more minor to cover only the interesting change. Therefore, we evaluate different RNN models to find the best window size or time step value. This parameter reflects the representation mechanism of the inputs. Precisely, what is the input length that characterizes the input change? It has a direct influence on the model performance. A small value may not represent the price change accurately, whereas a big number may contain more than a single change in price. We consider 6 different time steps (5, 10, 15, 20, 25, and 30 days) to do so. Note that we set the batch size and features using the previously evaluated values presented in Tables 6 and 7.

Tables 8 and 9 highlight the evaluation results of LSTM, GRU, BiLSTM, and BiGRU models for different Time Steps (TS) on Apple and Google Data, respectively. For Apple

Table 9 Evaluation of LSTM, GRU, BiLSTM, and BiGRU models for different Time Steps (TS) on Google Data

TS	LSTM			BiLSTM			GRU			BiGRU		
	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE
5	3.614	2.735	3.324	3.803	2.932	3.51	3.283	2.575	3.237	3.382	2.621	3.181
10	5.297	4.131	4.921	4.337	3.426	4.094	3.601	2.857	3.472	3.146	2.499	3.119
15	4.579	3.59	4.307	4.987	4.001	4.834	**3.194**	**2.492**	**3.408**	3.274	2.589	3.253
20	3.137	2.487	3.065	**2.422**	**1.903**	**2.431**	3.52	2.805	3.064	2.865	2.256	2.856
25	**2.95**	**2.289**	**2.816**	3.341	2.654	3.296	3.567	2.849	3.465	**2.46**	**1.908**	**2.455**
30	3.078	2.431	2.972	2.658	2.082	2.605	3.359	2.642	3.137	2.678	2.109	2.65

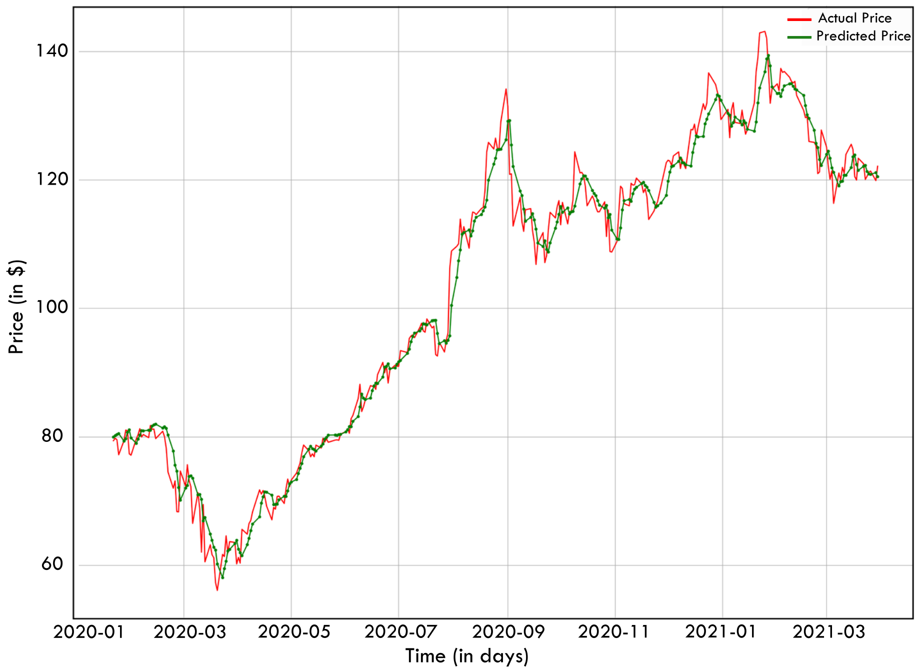

Fig. 8 Comparison of the actual close price and the predicted values for the BiGRU model for Apple data

date, the LSTM provides the best results using a window size of 10 days, GRU has the lowest error values using a window size of 25 days, while both BiLSTM and BiGRU showed small error values with a window size of 5 days with superiority of BiGRU. In general, the results provided by the evaluation metrics are close when using a window size of 5 to 15 days, and the values increase once they go above. On the other hand, for Google data, the LSTM records its lowest error values using a window size of 25 days, for which BiGRU records the lowest error values ever. BiLSTM and GRU have recorded their lowest error values using a window size of 20 and 15 days, respectively.

Figure 8 confirms the superiority of the BiGRU model among other RNN models on Apple data. Regarding forecasting accuracy, the model records the lowest RMSE value of 3.08, MAE of 2.31, and MAPE of 2.317. These values are given using a batch size of 128, a window size of 5 days, using dropout layers in the model's architecture, and finally, the F5 features: the open, high, low, and cumulative returns data. Figure 9 compares the actual close price and the predicted values for the BiLSTM model on Google Data. Using the same setting for the batch size, dropout, window size, and the F5 features, the BiLSTM model records the lowest RMSE value of 2.422, MAE of 1.903, and MAPE of 2.431.

As a result, the Bidirectional LSTM, Bidirectional GRU, and the ARIMAX models proposed in this paper perform closely in terms of error values. These models can reasonably predict the following day's closing price and provide better references and insights for future investments. Moreover, these performances are achieved using the F5 features.

Despite the abovementioned results, the problem of predicting stock prices remains an open challenge. The limitation of our proposal can be summarized as follows:

- The time series prices and technical indicators have to be separated and not concatenated in the same input vector. In other words, the adopted models have to expect the prices

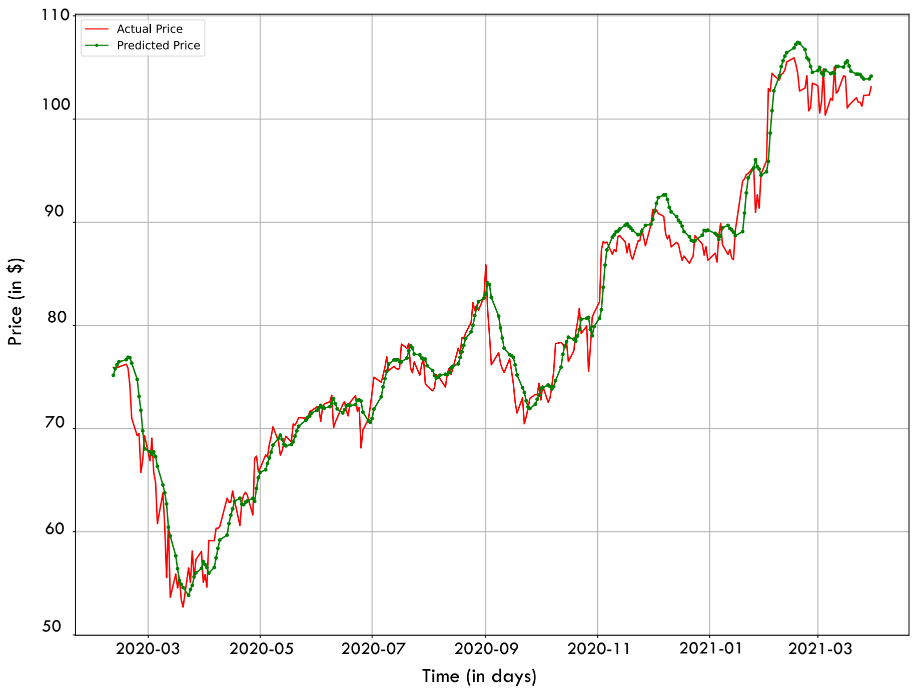

Fig. 9 Comparison of the actual close price and the predicted values for the BiLSTM model on Google Data

as input to catch the long and short-term dependencies before merging the outputs of the hidden state with the output of one or two layers treating the technical indicators. This will enhance a deeper price encoding with the exact window size to provide a solid prediction and indices for other applications.

- The transfer learning in our system is not straightforward. The developed model for a given stock works is not directly applicable to another stock with a different input scale. However, this is a general limitation for all automated trading applications.

5 Conclusion and future work

In this paper, a comparative study between different ML models to predict the closing price of the following day has been proposed. It consists of choosing several technical indicators besides the opening, highest, and lowest price as inputs for the models to enhance the input representation. In other words, these indicators work as a curiosity to drive the model in choosing accurate outputs. These relevant variables were selected for Apple's and Google stocks as an example to identify the effectiveness of technical indicators on the market regarding technical trading and predictive modeling. The experimental results have shown that the bidirectional recurrent neural networks and ARIMAX models presented the lowest forecasting errors regarding evaluation metrics and better performance than LSTM and GRU. Hence, we can state that the classical time series models, or ARIMA-based models, continue to perform well and are well suited for generating precise forecasts for financial time series.

However, even if these models made accurate predictions, they still have some limitations. Specifically, the models consider only the influence of historical data relating to the stock in question and do not incorporate opinion factors such as news and national politics. Thus, our future research will mainly incorporate the sentiment analysis side of stock news and national policies to ensure and make accurate predictions about the stock price. This mechanism will act as an additional curiosity mechanism to push the prediction and representation process forward.

Data availability Data sharing not applicable to this article as no datasets were generated or analyzed during the current study

Declarations

Conflict of Interest The authors declare that they have no conflict of interest.

References

1. Hirchoua B, Ouhbi B, Frikh B (2021) Deep reinforcement learning based trading agents: Risk curiosity driven learning for financial rules-based policy. Expert Syst Appl 170:114553. https://doi.org/10.1016/j.eswa.2020.114553
2. Kim S, Kang M (2019) Financial series prediction using attention lstm. arXiv:1902.10877 [cs, q-fin, stat]. Accessed 17 Sept 2022
3. Li W, Mei F (2020) Asset returns in deep learning methods: An empirical analysis on sse 50 and csi 300. Res Int Bus Financ 54:101291. https://doi.org/10.1016/j.ribaf.2020.101291
4. Bou-Hamad I, Jamali I (2020) Forecasting financial time-series using data mining models: A simulation study. Res Int Bus Financ 51:101072. https://doi.org/10.1016/j.ribaf.2019.101072
5. Guan H, Dai Z, Zhao A, He J (2018) A novel stock forecasting model based on high-order-fuzzy-fluctuation trends and back propagation neural network. PLoS ONE 13:0192366. https://doi.org/10.1371/journal.pone.0192366
6. Badr H, Ouhbi B, Frikh B (2020) Rules based policy for stock trading: A new deep reinforcement learning method. In: 2020 5th International conference on cloud computing and artificial intelligence: technologies and applications (CloudTech), pp 1–6. https://doi.org/10.1109/CloudTech49835.2020.9365878
7. Khan W, Ghazanfar MA, Azam MA, Karami A, Alyoubi KH, Alfakeeh AS (2020) Stock market prediction using machine learning classifiers and social media, news. J Ambient Intell Humaniz Comput. https://doi.org/10.1007/s12652-020-01839-w
8. Kabbani T, Duman E (2022) Deep reinforcement learning approach for trading automation in the stock market. IEEE Access 10:93564–93574. https://doi.org/10.1109/ACCESS.2022.3203697
9. Bao W, Yue J, Rao Y (2017) A deep learning framework for financial time series using stacked autoencoders and long-short term memory. PLoS ONE 12(7):1–24. https://doi.org/10.1371/journal.pone.0180944
10. Baek Y, Kim HY (2018) Modaugnet: A new forecasting framework for stock market index value with an overfitting prevention lstm module and a prediction lstm module. Expert Syst Appl 113:457–480. https://doi.org/10.1016/j.eswa.2018.07.019
11. Hung H-C, Chuang Y-J, Wu M-C (2021) Customizable and committee data mining framework for stock trading. Appl Soft Comput 105:107277. https://doi.org/10.1016/j.asoc.2021.107277
12. Zhou F, Zhou H-M, Yang Z, Yang L (2019) Emd2fnn: A strategy combining empirical mode decomposition and factorization machine based neural network for stock market trend prediction. Expert Syst Appl 115:136–151. https://doi.org/10.1016/j.eswa.2018.07.065
13. Krauss C, Do XA, Huck N (2017) Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the s&p 500. Eur J Oper Res 259(2):689–702. https://doi.org/10.1016/j.ejor.2016.10.031
14. Zhou F, Zhang Q, Sornette D, Jiang L (2019) Cascading logistic regression onto gradient boosted decision trees for forecasting and trading stock indices. Appl Soft Comput 84:105747. https://doi.org/10.1016/j.asoc.2019.105747
15. Thakur M, Kumar D (2018) A hybrid financial trading support system using multi-category classifiers and random forest. Appl Soft Comput 67:337–349. https://doi.org/10.1016/j.asoc.2018.03.006

16. Zhang D, Tang P (2023) Forecasting european union allowances futures: The role of technical indicators. Energy 270:126916. https://doi.org/10.1016/j.energy.2023.126916
17. Hirchoua B, Ouhbi B, Frikh B (2022) In: Ouaissa M, Boulouard Z, Ouaissa M, Guermah B (eds) The COVID-19 Pandemic's Impact on Stock Markets and Economy: Deep Neural Networks Driving the Alpha Factors Ranking, Springer, pp 219–243. https://doi.org/10.1007/978-3-030-77185-0_14
18. Hirchoua B, Mountasser I, Ouhbi B, Frikh B (2022) Evolutionary deep reinforcement learning environment: Transfer learning-based genetic algorithm. iiWAS2021, pp 242–249. Association for Computing Machinery. https://doi.org/10.1145/3487664.3487698
19. Tay FEH, Cao L (2001) Application of support vector machines in financial time series forecasting. Omega 29:309–317. https://doi.org/10.1016/s0305-0483(01)00026-3
20. Si Y-W, Yin J (2013) Obst-based segmentation approach to financial time series. Eng Appl Artif Intell 26:2581–2596. https://doi.org/10.1016/j.engappai.2013.08.015
21. Kumbure MM, Lohrmann C, Luukka P, Porras J (2022) Machine learning techniques and data for stock market forecasting: A literature review. Expert Syst Appl 197:116659. https://doi.org/10.1016/j.eswa.2022.116659
22. Henrique BM, Sobreiro VA, Kimura H (2018) Stock price prediction using support vector regression on daily and up to the minute prices. The Journal of Finance and Data Science 4:183–201. https://doi.org/10.1016/j.jfds.2018.04.003
23. Nayak RK, Mishra D, Rath AK (2015) A naïve svm-knn based stock market trend reversal analysis for indian benchmark indices. Appl Soft Comput 35:670–680. https://doi.org/10.1016/j.asoc.2015.06.040
24. Chen L, Qiao Z, Wang M, Wang C, Du R, Stanley HE (2018) Which artificial intelligence algorithm better predicts the chinese stock market? IEEE Access 6:48625–48633. https://doi.org/10.1109/access.2018.2859809
25. Jayanth Balaji A, Harish Ram DS, Nair BB (2018) Applicability of deep learning models for stock price forecasting an empirical study on bankex data. Procedia Comput Sci 143:947–953. https://doi.org/10.1016/j.procs.2018.10.340
26. Sezer OB, Gudelek MU, Ozbayoglu AM (2020) Financial time series forecasting with deep learning : A systematic literature review: 2005–2019. Appl Soft Comput 90:106181. https://doi.org/10.1016/j.asoc.2020.106181
27. Chalvatzis C, Hristu-Varsakelis D (2020) High-performance stock index trading via neural networks and trees. Appl Soft Comput 96:106567. https://doi.org/10.1016/j.asoc.2020.106567
28. Fischer T, Krauss C (2018) Deep learning with long short-term memory networks for financial market predictions. Eur J Oper Res 270(2):654–669. https://doi.org/10.1016/j.ejor.2017.11.054
29. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9:1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735
30. Gupta N, Jalal AS (2019) Integration of textual cues for fine-grained image captioning using deep cnn and lstm. Neural Comput Appl. https://doi.org/10.1007/s00521-019-04515-z
31. Yadav A, Jha CK, Sharan A (2020) Optimizing lstm for time series prediction in indian stock market. Procedia Comput Sci 167:2091–2100. https://doi.org/10.1016/j.procs.2020.03.257
32. Cho K, van Merrienboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, Bengio Y (2014) Learning phrase representations using rnn encoder–decoder for statistical machine translation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). https://doi.org/10.3115/v1/d14-1179
33. Xu J, Cai Y, Wu X, Lei X, Huang Q, Leung H-F, Li Q (2020) Incorporating context-relevant concepts into convolutional neural networks for short text classification. Neurocomputing 386:42–53. https://doi.org/10.1016/j.neucom.2019.08.080
34. Box GEP, Jenkins GM, Reinsel GC (2008) Time series analysis. Wiley Series in Probability and Statistics. https://doi.org/10.1002/9781118619193. Accessed 03 Dec 2019
35. Pankratz A (1991) Forecasting with dynamic regression models