Enhancing Forecasting Accuracy through Artificial Intelligence-Driven Complexity-Conscious Prediction Modeling

Raghurami Etukuru

Principal AI Scientist, AISciences.ai, Monroe Township, NJ 08831, USA.

Tel.: 17187025746; email: raghu.etukuru@aisciences.ai (R.E.)

Manuscript submitted January 12, 2024; accepted January 31, 2024; published February 19, 2024. DOI: 10.18178/JAAI.2024.2.1.60-78

Abstract: Traditional linear and simpler models often fail to capture the complex, multifaceted nature of real-world data, leading to inaccurate predictions. This research addresses this challenge by exploring the potential of complexity-conscious prediction, which seeks to incorporate the inherent intricacy within the data. The paper aims to demonstrate the significance of acknowledging and incorporating data complexity in forecasting models, especially in domains where accurate predictions are crucial for informed decision-making and can have a profound impact. By employing statistical methods to measure intricate patterns and by developing advanced deep learning models, such as Long Short-Term Memory (LSTM) networks and Generative Adversarial Networks (GANs), this research endeavors to achieve more accurate and reliable forecasts. Both LSTM and GAN models demonstrated remarkable capability in handling complex time series data, with MAPE values below 3.5%, indicating high accuracy. The GAN model, in particular, showed exceptional performance with a MAPE of less than 2% across all tested stocks, underscoring its advanced predictive capabilities. The findings suggest that deep learning models, especially GANs, substantially improve accuracy over traditional linear forecasting methods. This supports the thesis that integrating data complexity into predictive models through advanced deep learning techniques can significantly enhance forecast precision, thus providing a notable advantage in fields where accurate forecasting is crucial.

Keywords: AI-Driven forecasting, complexity-conscious prediction, forecasting accuracy, generative adversarial networks, intricate patterns, time series.

1. Introduction

In this study, we investigated the application of artificial neural networks for forecasting time series data characterized by intricate patterns. Traditional models for time series analysis often falter in accurately capturing these complex patterns. We aim to decode these intricate patterns and develop Artificial Intelligence (AI) models that effectively incorporate the data's complexity to enhance prediction accuracy. The complexity observed in these data patterns stems from multiple factors, each contributing a unique layer of difficulty. A primary factor is nonlinearity, indicating that the relationships among variables are not straightforward, often resulting in unpredictable and occasionally chaotic patterns. Another critical aspect is nonstationarity, where the statistical properties of the data, such as trends and seasonal variations, are in constant flux.

Additionally, the concept of long memory or dependence adds to this complexity, where historical data points significantly influence current values over extended periods. Asymmetry in data is also notable, implying that the statistical properties of the data are not uniformly distributed across time, presenting distinct challenges at various intervals. Finally, inherently random stochastic processes contribute to forming highly intricate and challenging-to-predict patterns.

In my book, "AI-Driven Time Series Forecasting: Complexity-Conscious Prediction and Decision-Making [1]", I introduced the concept 'Complexity-Conscious Prediction.' This method in predictive modeling recognizes and integrates the inherent complexity of data. It involves two key stages: assessing the complexity of input data and designing models tailored to this complexity. This strategy is particularly pertinent when data displays complex patterns and behaviors inadequately addressed by simpler predictive models. In addition, it is also essential to constantly evaluate model performance to detect data drift and adjust the model accordingly throughout the model's life.

Complexity-conscious prediction represents an advanced, nuanced methodology to comprehend and forecast temporal patterns in the context of time series forecasting. This approach comprehensively acknowledges the inherent complexities within time series data, encompassing aspects such as nonlinearity, nonstationarity, long-term memory, asymmetry, and stochasticity and their cumulative impact on data patterns. Diverging from traditional linear models that presume independence among observations, complexity-conscious methods strive to encapsulate these intricate patterns. To achieve precise forecasting, this research investigated sophisticated deep learning architectures like Long Short-Term Memory (LSTM) networks and Generative Adversarial Networks (GANs) and emphasized their meticulous fine-tuning. It prioritized the thorough fine-tuning of these architectures. Central to this approach is the strategic selection and optimization of various model parameters. These include the number of layers, activation functions, dropout rates, units, dense layers, choice of optimizers, loss functions, batch size, and the number of epochs. Particularly for the GAN model, the study involves choosing appropriate networks for the Generator and Discriminator and fine-tuning each Network's parameters. Such meticulous adjustments play a pivotal role in effectively harnessing and integrating the inherent complexity of the data within the predictive models, ultimately leading to enhanced accuracy.

1.1. Problem Statement

The primary challenge addressed in this research paper is the inadequacy of traditional linear models in time series forecasting, particularly their failure to effectively capture and interpret real-world data's complex, multifaceted nature. Despite widespread use, these conventional models often oversimplify data patterns, leading to inaccurate predictions and suboptimal decision-making in critical domains. This problem is exacerbated in fields where data exhibits highly intricate and dynamic behaviors, outpacing the capabilities of linear forecasting methods. The inadequacy of these traditional models in handling complex data patterns limits their effectiveness and reliability, especially in scenarios where precision in forecasting is vital. Consequently, there is a pressing need for an alternative approach that can more accurately reflect and leverage the inherent complexity within time series data. This research addressed this gap by proposing and validating the complexity-conscious prediction approach, utilizing advanced statistical and deep learning methods to enhance forecast accuracy and reliability.

1.2. Purpose of the Research

This research paper aimed to explore and validate the concept of complexity-conscious prediction within the field of time series forecasting. This approach is critical in addressing the limitations of traditional linear models, which often fail to capture the multifaceted and intricate nature of real-world data. The paper has demonstrated the significance of acknowledging and incorporating data complexity in forecasting models, especially in domains where accurate predictions are crucial for informed decision-making and can have a profound impact. By employing statistical methods to measure intricate patterns and by developing advanced deep learning models, such as Long Short-Term Memory (LSTM) networks and Generative Adversarial Networks (GANs), this research endeavored to achieve more accurate and reliable forecasts. The central thesis is that a deeper understanding and integration of data complexity into predictive models can significantly enhance the precision of forecasts, providing a substantial advantage over conventional forecasting methods.

2. Literature Review

The problem in forecasting stock prices using time series data is like the problem in signal processing, which is challenging due to high noise, non-stationarity, and nonlinearity [2]. While the traditional time series models help predict stock prices, they often fail to predict accurately due to bubbles and non-linearity in the historical data [3]. The factors contributing to the non-linearity in stock price fluctuation are also nonlinear [4]. The long memory occurs when the autocorrelation decays very slowly and remains significant persistently for a prolonged time [5]. Though the time series models evolved from basic Autoregression (AR) to the advanced Generalized Autoregressive Conditional Heteroskedasticity (GARCH) model, the forecasting problem persists. The issues associated with the GARCH model are asymmetry and lack of long-term memory. The GARCH model cannot perform well with asymmetry even though it can minimize the forecasting errors by considering prior ones. Since the negative shock can be more damaging than the positive shock, the GARCH model can result in severe errors in forecasting future volatility [6]. The GARCH model cannot effectively capture long-term dependency or long memory [7].

2.1. Artificial Neural Networks (ANN)

ANN algorithms are computationally intensive models that mimic the brain's neural network structure. They are nonlinear and nonparametric, capable of learning independently, with varying degrees of supervision, to perform tasks such as prediction, classification, and decision-making. Deep learning harnesses the synergy of artificial neural networks (ANNs) and computational capabilities such as GPU and TPU. ANNs have one edge over traditional models: their flexibility to model complex nonlinear relationships.

ANNs are structured into three main interconnected layers: the input, hidden, and output. Each layer consists of units known as nodes or artificial neurons. The input layer accepts data and transmits it to the subsequent hidden layer. Each hidden layer, in turn, processes the received information and passes it to the next hidden layer, if there is one, or directly to the output layer in the absence of additional hidden layers. The hidden layers imbue the Network with the capacity to analyze intricate nonlinear relationships, amplifying the Network's flexibility and computational power. Each artificial neuron within the Network comprises weighted inputs, known as synapses, and an activation function that calculates the output based on the input received. The ultimate result of these calculations is then transmitted as a singular output from each neuron.

ANNs can vary significantly based on their architecture, the number and arrangement of neurons, the type of connections between neurons, and the learning rules they employ. This diversity allows different types of ANN to be optimized for specific tasks or data types. Some common types of ANN include feedforward neural networks (FNNs), recurrent neural networks (RNNs) and variants of these, convolutional neural networks (CNNs) and variants of these, Generative Adversarial Networks (GAN), and autoencoders, each with unique advantages and typical applications.

2.2. Long Short-Term Memory (LSTM)

Recursive Neural Networks (RNN) models can handle complex sequential data, making them highly versatile across numerous applications. However, these networks come with challenges, including training difficulties due to vanishing and exploding gradients and the heavy computational resources they require, particularly when processing longer sequences. Problems like the vanishing and exploding gradients can destabilize training and lead to poor models.

The variation of RNN, an LSTM functions similarly to a standard RNN but also incorporates a mechanism akin to a conveyor belt, which can carry information across multiple time steps without processing it. This conveyor belt is regulated by gates controlling whether the information is discarded and replaced with the current input or preserved and carried forward to the next step, effectively enabling the LSTM to retain information over extended periods and alleviating the vanishing gradients problem.

LSTM networks, characterized by their chain-like architecture, feature a distinct module structure that differs significantly from traditional neural network layers. An LSTM module comprises several interacting components instead of a single layer. Within an LSTM cell, there are three main types of gates: the input gate, the forget gate, and the output gate. These gates function as regulatory checkpoints, controlling how information flows through the Network.

The forget gate plays a crucial role in managing the cell's internal state by evaluating both the previous hidden state and the current input. It produces outputs that determine how much of the existing information in the cell state is retained or discarded.

Subsequently, the input gate is responsible for updating the cell state with new incoming information. This gate involves a combination of components that decide which parts of the cell state are to be updated and also generates new candidate values to be added to the state.

Lastly, the output gate dictates what final output is generated from the cell. This process involves first determining which parts of the cell state are relevant for the output and then transforming these parts into a suitable format for the final output. This transformed output is modulated to ensure that only pertinent information, as determined by the output gate, is included in the final output from the cell.

LSTM networks hold several advantages, making them popular in handling complex sequential tasks. Primarily, LSTMs are adept at overcoming the vanishing gradient problem and effectively learning long-term dependencies. This capability allows them to maintain their state over extended sequences, which proves highly beneficial in scenarios such as time series forecasting and NLP. LSTMs also exhibit flexibility as they can efficiently process inputs of varying lengths, adapting seamlessly to changes in sequence lengths across different data batches. This adaptability is particularly useful in tasks like text translation, where the length of the input sequence can significantly vary.

LSTM networks' inherent ability to retain prior information using their cell state makes them an advantageous choice for tasks where temporal dynamics are critical. Furthermore, much like other neural networks, LSTMs can be trained end-to-end using standard backpropagation techniques. Given enough data and computational resources, this enables them to learn complex tasks directly from raw input data. LSTMs also showcase robustness against noise, as their ability to discern between essential and discardable information allows them to cope effectively with irrelevant input data. Finally, the modularity of the LSTM design, enabling the stacking of cells into layers in a neural network, allows these models to be adapted to different architectures, facilitating the modification of the model's complexity as per requirements.

2.3. Generative Adversarial Network (GAN)

GANs are a class of ANN frameworks and consist of two neural networks, a generator and a discriminator, that are trained together in a competitive process. The Generator's goal is to create synthetic or fake data that resembles real data as closely as possible by learning the underlying data distribution of the training set. The Discriminator aims to distinguish between real data from the training set and fake data generated by the Generator. During training, the Generator tries to improve its generated data to fool the Discriminator better, while the Discriminator tries to improve its ability to correctly identify real and fake data. This adversarial process continues until an equilibrium is reached, where the Generator produces data nearly indistinguishable from the real data, and the Discriminator can no longer differentiate between the two. GANs have shown impressive results in various applications, such as image synthesis, image-to-image translation, data augmentation, and style transfer.

The Generator (G) is a function that takes a random noise vector (z) and maps it to data-space ($G(z)$). The purpose of the Generator is to estimate the distribution that the real data comes from to generate new data from that same distribution. The Discriminator (D) is a binary classification network that inputs data (x) and outputs a scalar ($D(x)$), representing the probability that x came from the real data rather than the Generator.

The objective function of a GAN describes the competition between the Generator and Discriminator. It is formulated as a two-player minimax game with the following loss function:

$$\min G \max D\ V(D, G) = E_{x\sim Pdata(x)}[\log D(x)] + E_{z\sim Pz(z)}[\log(1 - D(G(z)))].$$

Here, E represents the expectation, $Pdata(x)$ is the real data distribution, and $Pz(z)$ is the distribution of the input noise vectors. The first term in the equation corresponds to the Discriminator's ability to recognize real data, and the second term corresponds to its ability to recognize fake data. The Generator aims to minimize this function, while the Discriminator aims to maximize it.

In a real application, the sample generator is typically implemented as a neural network, often a CNN or FFN. The architecture and complexity of the generator network can vary depending on the specific application and complexity of the generated data. The discriminator network is also typically implemented as an ANN, often a CNN or a FFN, LSTM, or GRU. The training process involves both the Generator and Discriminator being trained simultaneously. The Discriminator is trained by feeding it both real and fake data, and it updates its weights through backpropagation using the labels (real for real data, fake for data from the Generator). The Generator is then trained using the output of the Discriminator. It wants the Discriminator to believe its output is real, so it uses its predictions to update its weights.

During the training process, both networks improve by adjusting their internal parameters. This is done using gradient descent, a method of optimization that iteratively adjusts parameters in the direction that minimizes the loss function. The specific updates depend on the gradients of the loss function with respect to the parameters, which are computed using backpropagation.

3. Materials and Methods

The data required for the research was collected from Yahoo Finance. The stock data includes a history of observations from January 2, 2008, to November 13, 2023. The variables identified in the stock quotes dataset include a date, open price, high price, low price, close price, and volume. The description of the variables and their type are specified in Table 1.

Table 1. Description of the Variables

Variable	Description	Data Type
Ticker	Symbol of the Stock	Text
Date	Date of the record	Date
Open Price	The open price is the price at which the first trade is executed when the stock exchange opens	Numeric
Stock Ticker	The symbol of the stock	Text
High Price	The high price is the maximum price the stock trades intraday	Numeric
Low Price	The low price is the minimum price for the trade during intraday	Numeric
Close Price	The close price is the price at which the last trade is executed before the stock	Numeric
Volume	Total volume traded on the record date	Numeric

This study ensured that the time series data has complex patterns of nonlinearity, nonstationarity, long-term memory, asymmetry, and stochasticity by using appropriate statistical methods. Since the historical data have daily observations, the price fluctuations that occurred throughout the day were ignored.

Instead, the prices at four levels, open, high, low, and close, were used to forecast the next day's closing price.

The time series data for each stock was divided into two sets; the first part was used to train and optimize the model, and the second data set was used to test the model on an 80:20 basis. In other words, the first part is the training data set, and the second is the test data set. In cases where there is no dependence from one observation to another, the data can be randomly split on an 80:20 basis. Since, in time series data, one observation can influence the following observation, the data cannot be divided randomly. Therefore, the values at the frontal 80 percent of the time series were used for training, and the rear 20 percent were used for testing.

The data was normalized before the time series data was used to train and test the AI models. The data normalization method produces high-quality data that improves the effectiveness of the learning algorithm [8]. The Min-Max Normalization method was used to normalize the time series data. The Min-max normalization method scales a variable in the sample set to [-1, 1] or [0, 1] [9].

Models were developed with careful selection and optimization of numerous parameters, including the number of layers, activation functions, units, dense layers, dropout rate, optimizers, loss functions, batch size, and epochs. Such detailed adjustments are crucial for effectively capturing and integrating the intrinsic complexity of the data into the predictive models, thereby enhancing accuracy.

The frontal part of the dataset was fed into the model. The model then learns to relate the input to the output and optimizes it. The model was then provided with the second part of the data set. The model generated the outcome based on the input in the data set. The statistical measure, mean absolute percentage error (MAPE), was used to measure the model's accuracy. The lower values of MAPEs indicate that the model is more accurate. In other words, the lower the MAPE, the higher the accuracy. MAPE is also helpful in comparing the accuracy of different forecasting models, as it indicates how significant the errors are concerning the actual values. Alternatively, the root mean squared error (RMSE) could have been used to measure accuracy. However, using the RMSE makes it challenging to assess accuracy as the RMSE is a relative measure, and it only measures the magnitude of the error between the predicted and actual values.

3.1. Quantifying Intricate Patterns

Complex patterns are classified into deterministic, stochastic, and mixed. Deterministic intricate patterns arise from nonlinear deterministic systems characterized by complex and chaotic behavior. Despite following specific rules, these systems can appear random due to their sensitivity to initial conditions and nonlinearity. A well-known example is the weather system, which adheres to deterministic physical laws but exhibits complex and seemingly unpredictable patterns due to its chaotic nature. On the other hand, complex stochastic patterns occur when a random process generates data. Even simple stochastic processes can produce intricate patterns. For instance, a random walk, a basic stochastic process, generates a complex, nonrepeating pattern that can be challenging to differentiate from a deterministic chaotic process. Financial market data, including stock prices, is often modeled as stochastic processes and can exhibit complex patterns. Lastly, mixed intricate patterns are observed in real-world scenarios where time series data combines deterministic and stochastic elements. An example is the El Niño phenomenon, a climate pattern involving deterministic components governed by physical laws and stochastic factors stemming from random disturbances. Statistical methods can be applied to quantify the complex patterns observed in data, such as nonstationarity, nonlinearity, long memory or dependence, asymmetry, and stochasticity.

3.2. Measuring Nonstationarity

Nonstationarity in time series data implies the statistical properties of a sequence of observations change over time. These statistical properties include mean, variance, and autocorrelation measures. Nonstationary processes contrast with stationary processes, where these properties are constant over time. Nonstationarity can manifest in trend, unit root, or heteroskedasticity.

The Augmented Dickey-Fuller (ADF) test, a variant of the Dickey-Fuller test, is a statistical tool used primarily to assess the nonstationarity of historical data and ascertain the stationarity of a time series. It checks for the existence of a unit root in a time series to distinguish between stationary and nonstationary time series. The ADF test tests the null hypothesis that a time series has a unit root (i.e., is nonstationary). If the test rejects the null hypothesis, it suggests the time series is stationary. If it fails to reject the null hypothesis, it indicates the time series has a unit root and is nonstationary. The p-value obtained from the test interprets the stationarity, where a value less than 0.05 signifies stationarity, while a value exceeding 0.05 implies nonstationarity of the time series.

3.3. Measuring Nonlinearity

Nonlinearity in time series data refers to situations where changes in the series' output are not directly proportional to input changes. In other words, the relationship between past and future values of the series is not a straight line, and small changes in the input can lead to disproportionally large changes in the output or vice versa. Nonlinear time series display diverse and intricate behaviors, including cycles that vary in length, amplitude changes, and chaotic dynamics. Unlike linear sine or cosine waves with constant cycle lengths, nonlinear time series can exhibit cycles that fluctuate in duration over time. The amplitude, representing the size of these cycles, may also change, resulting in increased or decreased volatility within the series.

Additionally, certain nonlinear systems can exhibit chaotic behavior, whereby the time series appears random but is governed by deterministic rules. Chaotic systems are susceptible to initial conditions, causing even minor differences in starting values to yield vastly divergent outcomes, rendering long-term prediction nearly impossible. Nonlinear models can be challenging to estimate and interpret, as they often have multiple parameters and may include complex interactions between variables. They can also be more prone to overfitting than linear models.

The Lyapunov test can scrutinize the stability of a dynamical system and the nonlinearity within the time series data. This test utilizes the Lyapunov exponent, a measure that quantifies the divergence rate of trajectories in close proximity within a dynamic system. This indicates the degree of nonlinearity in the time series. If the Lyapunov exponent is positive, it suggests the system is in a state of chaos, implying that trajectories close to each other will exhibit exponential divergence as time progresses. Consequently, even minimal variances in the starting conditions can drastically alter long-term outcomes.

3.4. Measuring Long-Term Memory

Long-term memory in time series data refers to the dependence of current observation on historical observations from a significant distance in the past. This characteristic goes beyond what is typically observed in many standard time series models like AutoRegressive (AR) or Moving Average (MA), where the dependencies usually exist over a short or fixed period. In long-memory processes, these dependencies decay more slowly and can extend far into the past.

When a time series displays long-term memory, it retains a certain degree of persistence or memory of its past values. If the series experienced a high value in the past, it is more likely that future values will also be high and, similarly, for low values. This dependency can extend over surprisingly long periods, hence the term "long-term memory." This feature is particularly prevalent in various natural and social phenomena. For example, in finance, the volatility of financial returns can show long memory, meaning that periods of high volatility tend to be followed by similarly volatile periods, and the same for low volatility.

The detrended fluctuation analysis (DFA) technique is employed to ascertain the presence of long-term memory within the time series. This method is primarily utilized to determine the long-term correlations or fluctuations within a time series signal. The Hurst exponent, an outcome of the DFA method, helps quantify the correlation traits of a time series. If the Hurst exponent exceeds 0.5, it implies the time series exhibits a positive correlation and displays long-term persistence. Conversely, a Hurst exponent below 0.5 signifies the time series is anti-persistent, indicating smaller ones likely follow larger fluctuations. A Hurst exponent is precisely equal to 0.5, denoting that the time series lacks any correlational properties.

3.5. Measuring Asymmetry

Asymmetry in time series data refers to when the statistical properties of the data are not the same across different levels or periods. This contrasts with symmetric time series, where the properties of the data remain the same regardless of time shifts. There are several ways in which asymmetry can manifest in time series data. Nonlinear dependencies in a time series can lead to asymmetries. For example, a series might exhibit more substantial dependencies or more extreme values in one direction than another.

In financial time series, it is common to observe that negative shocks, like a decrease in price or return, lead to a greater increase in volatility (risk) than positive shocks of the same magnitude, a phenomenon often referred to as the "leverage effect." In some time series, the seasonal patterns may not be symmetric. For example, in a series of monthly sales data for a retail store, the sales increase leading up to the December holiday season may be more rapid than the decrease afterward.

Skewness is a statistical measure that can quantify the degree of asymmetry of a probability distribution and, by extension, a time series. In a time series context, skewness can be used to analyze whether the distribution of the data points in the series is symmetric (having a skewness close to 0) or if it leans toward one side, either to the right (positive skewness) or the left (negative skewness). If a time series has a positive skewness, the distribution's right tail is longer or fatter than the left one. Practically, this means that exceptionally large values are more likely than exceptionally small ones. For instance, a positive skew in a daily sales data time series might indicate occasional days with very high sales, but days with very low sales are less frequent or extreme. On the other hand, if a time series has a negative skewness, it means the left tail of the distribution is longer or fatter than the right one. This would mean that the series has occasional exceptionally low values, but the high values are less extreme or less frequent.

Understanding the skewness of a time series can be very important in forecasting and modeling. Many time series models assume the errors are normally distributed, implying zero skewness. If the skewness is significantly different from 0, these models may not be appropriate, and data transformations or different models may be needed.

3.6. Measuring Stochasticity

Stochasticity refers to the randomness or unpredictability inherent in a process. A stochastic time series determines the future state not only by the current state but also by an element of randomness. In other words, the evolution of a stochastic time series is probabilistic. For example, consider the daily closing prices of a stock in the financial markets. Each day's closing price (tomorrow's price) depends on today's price and many other factors that introduce randomness, such as news about the company, economic indicators, or geopolitical events. We can model this using a stochastic process because the future stock price, while related to the current price, also involves a degree of unpredictability.

In the analysis of a stochastic time series, various elements require attention. The trend, for instance, represents the long-term direction in which the series is moving, demonstrating a gradual increase, decrease, or possibly no change. Seasonality, another critical feature, denotes regularly repeating patterns in the data occurring at fixed time intervals, such as daily, monthly, or annually. This aspect accounts for time-driven events like seasonal sales spikes in retail. Cycles, in contrast to seasonality, refer to patterns in the data not confined to a fixed length. These can embody fluctuating phenomena like business or economic cycles lacking a defined period. Finally, there are random or irregular movements. These changes in the time series can't be attributed to trend, seasonality, or cycles and are often considered noise or the random error component of the series. These irregularities contribute to the stochasticity, or unpredictability, inherent in the time series.

Entropy can provide a measure of the predictability of the time series. Entropy is a concept that originated in thermodynamics and information theory, which can be used to quantify the level of uncertainty, disorder, or randomness, in this context, the stochasticity, in a dataset. A lower entropy suggests a more predictable (less stochastic) time series, while a higher entropy corresponds to more disorder or randomness (greater stochasticity). In a perfectly deterministic system, entropy would be zero because future states can be predicted with certainty from current states. While variations of entropy exist, Shannon entropy is widely used to measure uncertainty.

The equation for Shannon entropy is $H(X) = - \Sigma\, p(x) \log2\, p(x)$.
where $p(x)$ represents the probability of an event x, the sum is over all possible events.

If the calculated entropy exceeds 90 percent of the maximum possible entropy, then it is said that the time series is stochastic. This maximum entropy was determined as the logarithm of the number of unique elements within the time series data, that is, log(len(np.unique(time_series_data))). The high entropy indicates that the time series is highly unpredictable, indicating a high degree of stochasticity.

Table 2. Statistical Properties of Time Series

Stock Symbol	**Non-Stationarity (Augmented Dickey-Fuller (ADF) p-value)**	**Non-Linearity (Lyapunov Exponent)**	**Long-Term Memory (Detrended Fluctuation Analysis (DFA) Hurst Exponent)**	**Asymmetry (Skewness)**	**Stochasticity (Entropy (Maximum Entropy))**
UNH	0.99	0.002	1.46	0.89	8.17 (8.21)
ATO	1.43	0.002	1.43	0.14	7.92 (8.01)
FAST	0.92	0.003	1.46	0.79	7.90 (7.99)
NI	0.72	0.001	1.47	-0.18	7.57 (7.74)
COST	0.99	0.002	1.50	1.07	8.17 (8.20)

Table 2 shows the analysis of the patterns in the time series of the stocks using a suite of statistical metrics revealing critical characteristics of their time series behavior. These metrics include the Augmented Dickey-Fuller (ADF) p-value for assessing stationarity, the Lyapunov exponent for gauging nonlinearity, the Detrended Fluctuation Analysis (DFA) Hurst exponent for evaluating long-term memory, skewness for measuring asymmetry, and entropy (and maximum entropy) concerning its maximum possible value for determining stochasticity.

From the analysis, a p-value greater than 0.05 in the ADF test indicates nonstationarity, suggesting that the statistical properties of the time series, such as mean and variance, are not constant over time. A positive Lyapunov exponent suggests the presence of chaos within the system, characterized by an exponential divergence of initially close trajectories over time. This finding highlights the nonlinearity inherent in the time series.

When exceeding 0.5, the Hurst exponent suggests a positive long-term correlation, indicating persistence in the time series. This means that trends observed in the past are likely to continue in the future. Skewness, whether positive or negative, indicates asymmetry in the time series. Positive skewness implies a longer right tail in the distribution, whereas negative skewness indicates a longer left tail.

Finally, entropy values exceeding 90% of the maximum possible entropy denote a high level of stochasticity in the time series. This high entropy level suggests significant unpredictability and randomness, further underscoring the complexity of forecasting these stock time series.

The complete code used to measure patterns is available on GitHub [10]. For more details, please visit https://github.com/raghu-etukuru/Complexity-Conscious-Prediction." Fig. 1 to Fig. 5 visualize the time series data of the stocks used in the research.

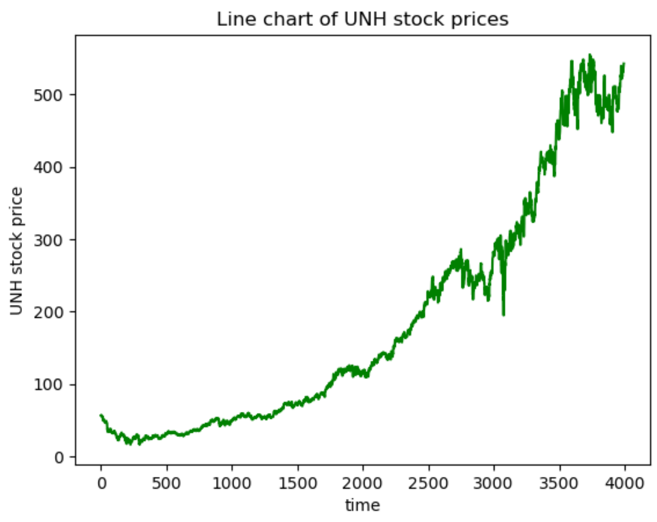

Fig. 1. Time series of the stock UNH.

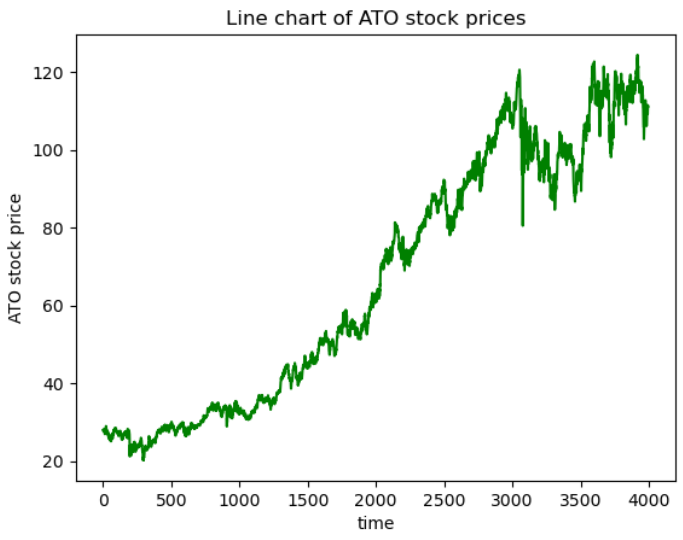

Fig. 2. Time series of the stock ATO.

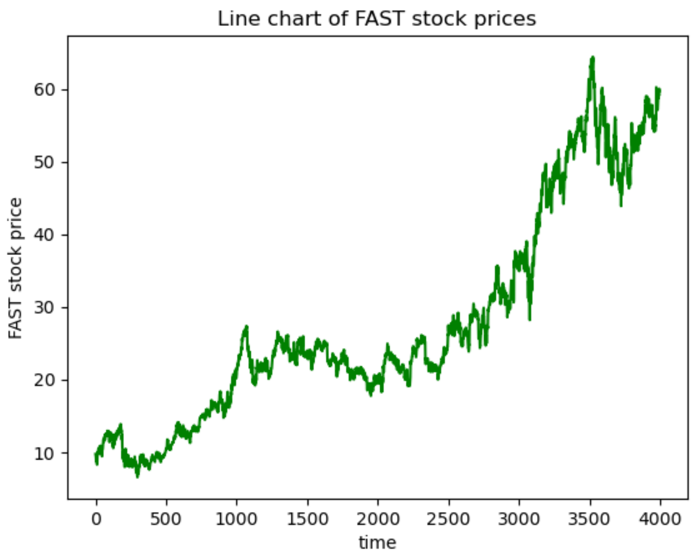

Fig. 3. Time series of the stock FAST.

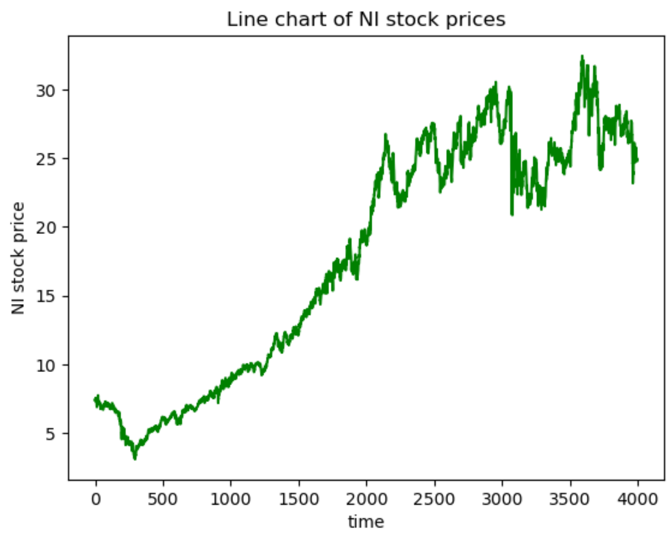

Fig. 4. Time series of the stock NI.

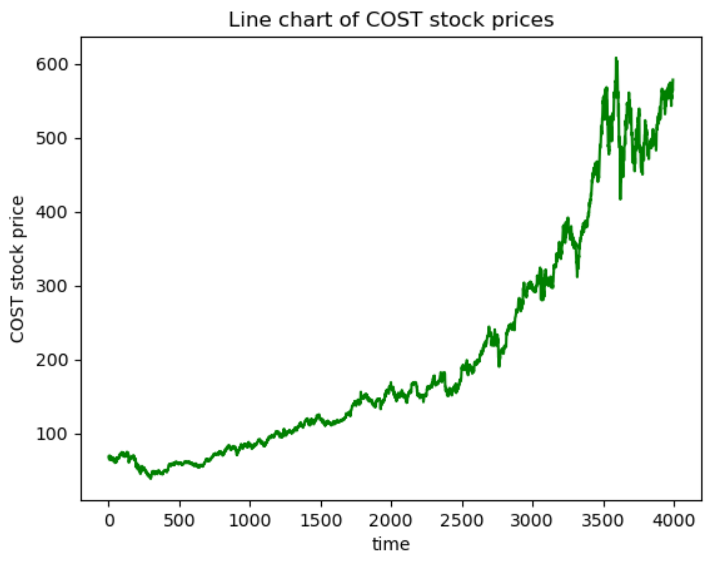

Fig. 5. Time series of the stock COST.

4. Design and Settings

4.1. Long Short-Term Memory

We have incorporated a Long Short-Term Memory (LSTM) model, known for its efficacy in handling time series data. The model is constructed using a Sequential framework, enabling a systematic layer-by-layer assembly. The primary layer is an LSTM unit with 512 nodes, selected for its capacity to process the input data characterized by a specified number of time steps (input_dim) and features (feature_size). This layer utilizes the 'tanh' activation function due to its proficiency in managing data nonlinearities. A critical feature of this layer is the return_sequences = True parameter, which allows the output from each time step to be passed to the next layer, preserving the temporal sequence of the data. A Dropout layer with a 20% dropout rate is introduced following the first LSTM layer to mitigate the risk of overfitting. This strategy randomly omits a portion of neural connections during training, enhancing the model's ability to generalize to unseen data.

Further refining the architecture, a second LSTM layer with 384 units is employed, focusing on the final output of the sequence, thus condensing the temporal information. After this, the LSTM layer is two Dense layers. The first Dense layer comprises 64 units, serving as a transition from the high-dimensional output of the LSTM to a more compressed form. The final layer is a Dense unit with a single output node, aligning with the typical requirement of time series forecasting where the output is generally a singular continuous value. The model is compiled using the 'Adam' optimizer because it effectively handles sparse gradients. Mean Squared Error (MSE) is utilized for loss calculation, aligning well with the regression nature of time series forecasting. The training process involves 25 epochs with a batch size of 25, where the model's weights are iteratively adjusted to minimize the loss function. This process significantly enhances the model's ability to accurately predict future values based on learned patterns from the training data. In the final phase, the trained model generates predictions on the test dataset, providing a crucial evaluation of the model's forecasting capabilities. The complete code of this LSTM model is available on GitHub[10]. For more details, please visit https://github.com/raghu-etukuru/Complexity-Conscious-Prediction." The following is an excerpt from the code.

```
model = Sequential()
model.add(LSTM(units = 512, return_sequences = True, activation = 'tanh', input_shape = (input_dim,
feature_size)))
model.add(Dropout(0.2))
model.add(LSTM(units = 384, activation = 'tanh'))
model.add(Dense(64))
model.add(Dense(units=1))
model.compile(optimizer='adam', loss='mse')
model.fit(X_train, y_train, epochs=25, batch_size = 25)
yhat = model.predict(X_test, verbose=0)
```

4.2. Generative Adversarial Network

In the design of the GAN for time series forecasting, two critical components are developed: the Generator and the Discriminator. The Generator model is designed using a Sequential model architecture. It employs a Gated Recurrent Unit (GRU) layer, crucial for handling time series data. This layer has 1024 units and uses a 'tanh' activation function. It is designed to process input with a defined number of time steps (input_dim) and features (feature_size).

Following the GRU layer, the model includes a series of Dense layers. These layers progressively decrease in size: 512, 256, 128, and 64 units. This design choice creates a funnel effect, where the Network condenses the information learned from the time series data. The final layer of the Generator is a Dense layer with a unit size equal to the output_dim, representing the dimensionality of the generated time series output. The complete code of this GAN model is available on GitHub[10]. For more details, please visit https://github.com/raghu-etukuru/Complexity-Conscious-Prediction." The following is an excerpt from the code.

```
def create_generator_model():
        gen_model = Sequential()
        gen_model.add(GRU(units = 1024, activation = 'tanh', input_shape = (input_dim, feature_size)))
        gen_model.add(Dense(512))
        gen_model.add(Dense(256))
        gen_model.add(Dense(128))
        gen_model.add(Dense(64))
        gen_model.add(Dense(units=output_dim))
        return gen_model
```

The Discriminator is also built using a Sequential model architecture but differs significantly from the

Generator. It employs a 1D Convolutional (Conv1D) layer. This layer is designed to process the time series data with 32 filters, a kernel size of 5, and a stride of 2. It uses 'relu' activation and is set to have padding 'same' to maintain the input size.

Following the Conv1D layer, the Discriminator employs a series of Dense layers with 'relu' activation functions and no biases. The units in these layers are 230 and 220, respectively, allowing the model to process further and learn from the features extracted by the Conv1D layer. The final layer is a Dense layer with a single unit and a 'sigmoid' activation function. This design choice is typical for binary classification tasks, which, in the case of GANs, involves distinguishing between real and generated (fake) time series data.

```
def create_discriminator_model():
    disc_model = Sequential()
    disc_model.add(Conv1D(32, input_shape=(n_steps_in+1, 1), kernel_size=5, strides=2,
padding='same',
                          activation='relu'))
    disc_model.add(Dense(230, use_bias=False, activation='relu'))
    disc_model.add(Dense(220, use_bias=False, activation='relu'))
    disc_model.add(Dense(1, activation='sigmoid'))
    return disc_model
```

The design of both the Generator and Discriminator reflects a deep learning approach tailored for time series data. The Generator focuses on generating realistic time series data, while the Discriminator is optimized to differentiate between real and generated data. The architecture choices, such as the use of GRU and Conv1D layers, are particularly suited for capturing the temporal dynamics and patterns inherent in time series data.

In our research on forecasting with time series data, we use a training method that's a bit like a teaching session between two students: the 'generator' and the 'discriminator.' Imagine the Generator as a student trying to create realistic-looking but fake time series data. It learns by taking real data and trying to replicate it. On the other hand, the Discriminator is like a student who learns to tell the difference between the real data and the fake data created by the Generator.

During training, the Generator creates data and presents it to the Discriminator. The Discriminator also looks at real data and tries to figure out which is real and which is fake. Both the Generator and the Discriminator are learning from this process: the Generator is learning to make more realistic data, and the Discriminator is getting better at telling real from fake. As they learn, we measure how well each is doing through 'loss.' If the Discriminator is easily fooled, its loss is high; if it's good at telling real from fake, its loss is low. The same goes for the Generator. Based on these measurements, the Generator and the Discriminator adjust their strategies to improve.

Over time, this process leads to a generator that creates very realistic time series data and a discriminator that's very good at telling the difference between real and generated data. This method is crucial for our research, as it helps us understand and predict time series data more accurately.

5. Results

The mean absolute percentage error (MAPE) was used as a benchmark to measure the forecasting accuracy of each stock. Since the MAPE calculates the average of the absolute differences between the actual and predicted values, expressed as a percentage of the actual values, it is straightforward to interpret [11]. MAPE is also helpful in comparing the accuracy of different forecasting models, as it indicates how significant the errors are concerning the actual values. Table 3 shows the MAPE values of the stocks produced by the GAN and LSTM models.

Table 3. Accuracy of Forecasting

Stock Symbol	GAN MAPE	LSTM MAPE
NI	1.37	1.28
COST	1.45	1.93
ATO	1.50	1.88
UNH	1.52	3.27
FAST	1.72	2.64

The model with a MAPE of less than ten is considered a highly accurate prediction (HAP) model, and with a MAPE of between 10 and 20 is regarded as a good predictive model [12]. Our research and comparative analysis revealed that both LSTM and GAN models can handle complex time series data. A critical finding was their predictive accuracy, which was measured using MAPE. Both models achieved a MAPE of less than 3.5%, indicating very high accuracy for all the stocks in the study. Notably, the GAN model demonstrated exceptional performance, with a MAPE of less than 2% for all stocks in the study. The lower the MAPE, the more accurate the model, thus highlighting the superior predictive capability of the GAN model in this context. Fig. 6 to Fig. 15 show the comparison of actual vs predicted prices.

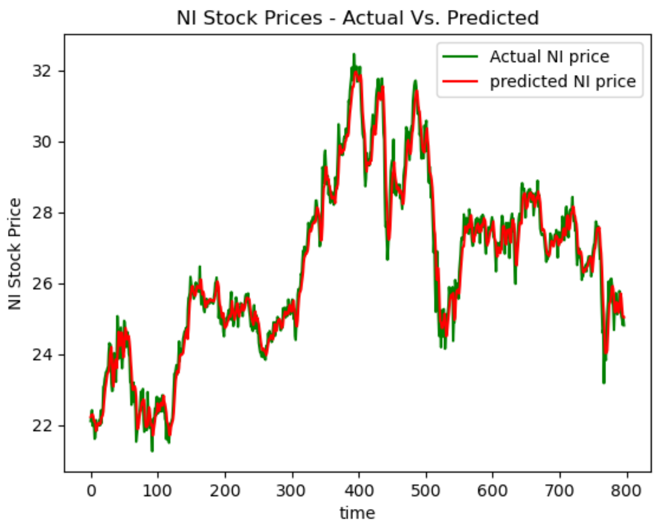

Fig. 6. Actual time series and LSTM predicted time series for the stock NI.

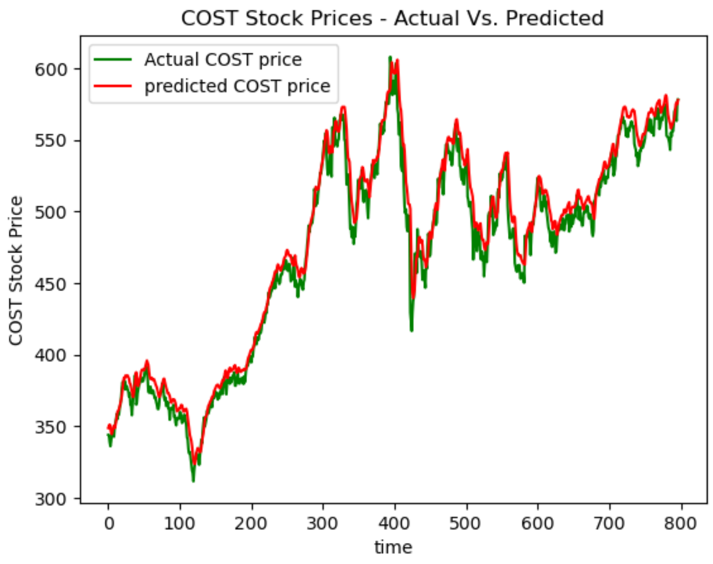

Fig. 7. Actual time series and LSTM predicted time series for the stock COST.

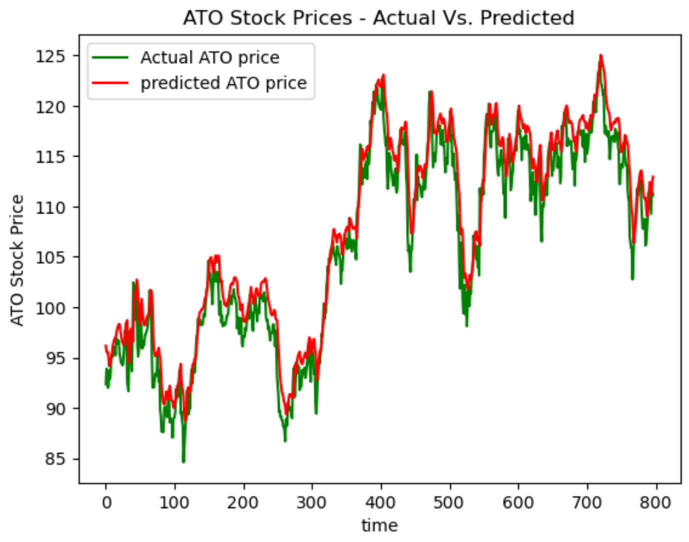

Fig. 8. Actual time series and LSTM predicted time series for the stock ATO.

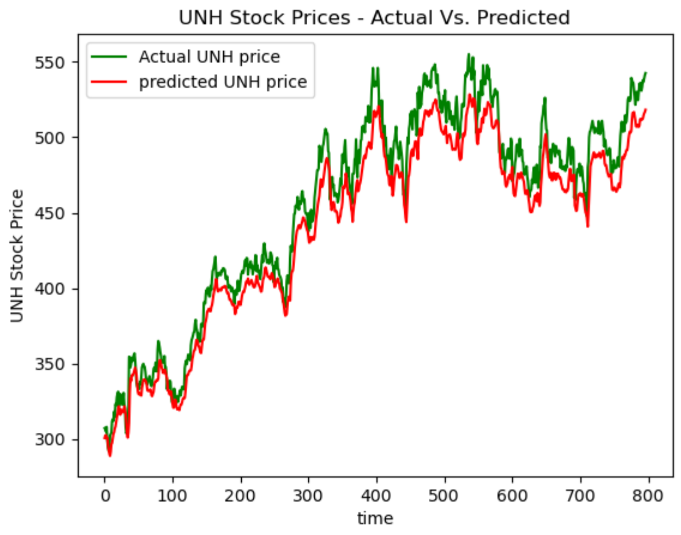

Fig. 9. Actual time series and LSTM predicted time series for the stock UNH.

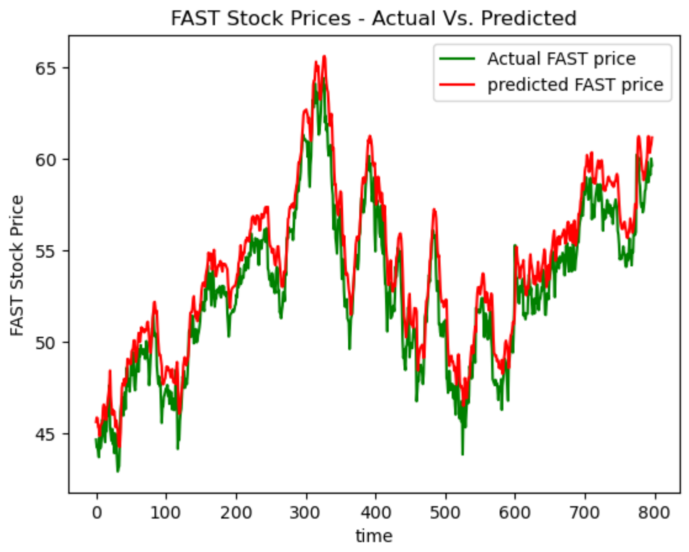

Fig. 10. Actual time series and LSTM predicted time series for the stock FAST.

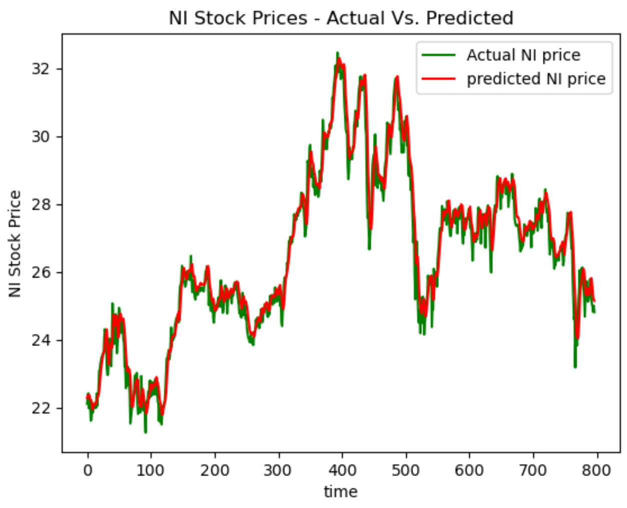

Fig. 11. Actual time series and GAN predicted time series for the stock NI.

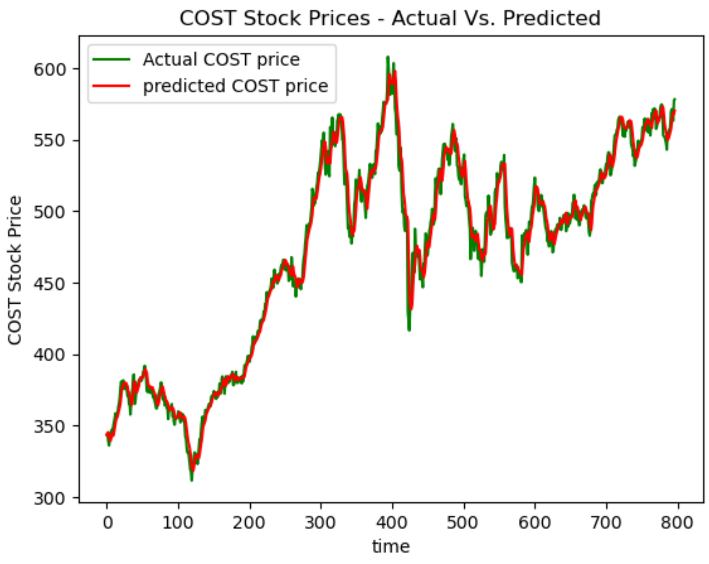

Fig. 12. Actual time series and GAN predicted time series for the stock COST.

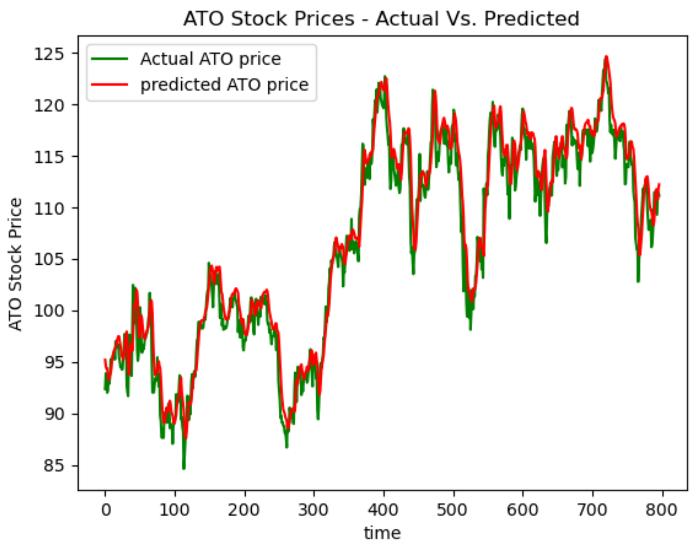

Fig. 13. Actual time series and GAN predicted time series for the stock ATO.

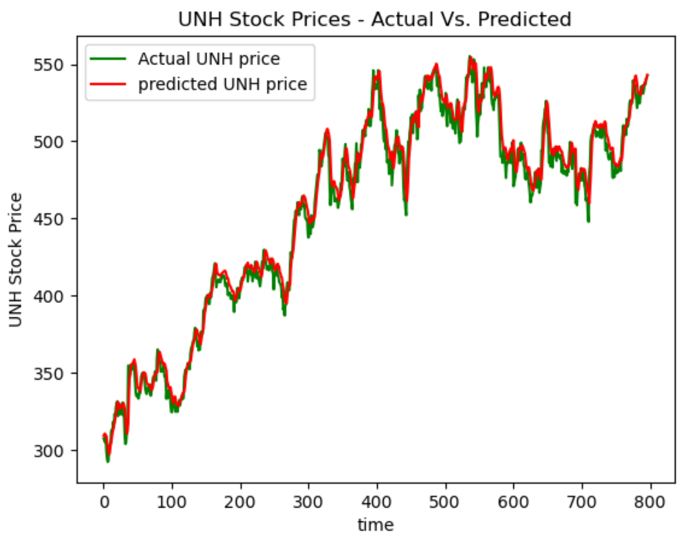

Fig. 14. Actual time series and GAN predicted time series for the stock UNH.

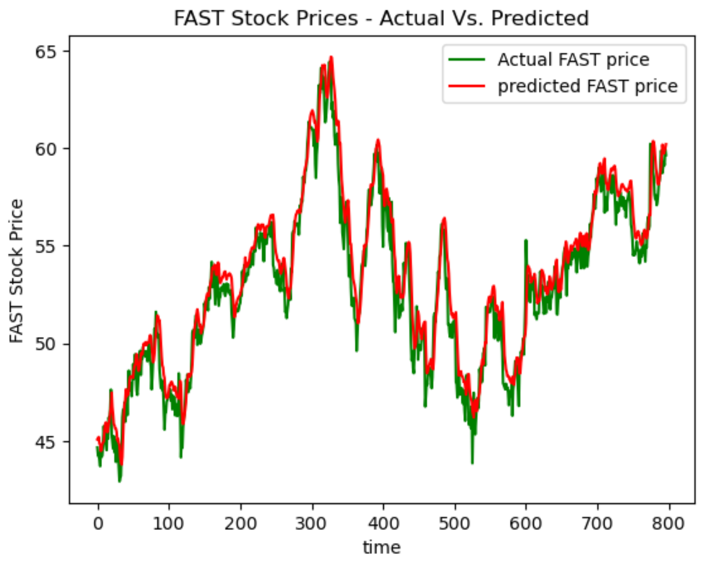

Fig. 15. Actual time series and GAN predicted time series for the stock FAST.

6. Discussion

Complexity-conscious prediction in artificial intelligence (AI) refers to a predictive modeling approach that actively acknowledges and incorporates the inherent complexity in data. This method in predictive modeling recognizes and integrates the inherent complexity of data. It involves two key stages: assessing the complexity of input data and designing models tailored to this complexity. This approach is especially relevant in fields where data exhibits intricate patterns and behaviors that traditional, simpler predictive models cannot adequately capture. This research focused on the realm of complexity-conscious prediction, with a specific lens on time series forecasting.

The inherent complexity within time series data included features like nonlinearity, nonstationarity, long-term memory, asymmetry, and stochasticity. Traditional linear models often struggle to capture these complex interrelationships due to their simplistic assumption of independence among observations. To address this, our study utilized statistical methods to measure the complexity of time series data and employed advanced deep learning architectures adept at interpreting these multifaceted characteristics.

The primary techniques under examination were Long Short-Term Memory (LSTM) and Generative

Adversarial Networks (GAN). LSTM is particularly known for its capability to recognize long-term dependencies, which is essential for time series forecasting, where past data significantly influences future predictions. With its innovative generator-discriminator framework, GAN excels in generating accurate and realistic data sequences.

Our research and comparative analysis revealed that both LSTM and GAN models can handle complex time series data. A critical finding was their predictive accuracy, measured using the MAPE. Both models achieved a MAPE of less than 3.5%, indicating high accuracy for all the stocks in the study. Notably, the GAN model demonstrated exceptional performance, with a MAPE of less than 2% for all stocks in the study. The lower the MAPE, the more accurate the model, thus highlighting the superior predictive capability of the GAN model in this context.

These findings highlight the importance of adopting complexity-conscious methods in time series forecasting. By leveraging the sophisticated capabilities of LSTM and GAN, more accurate and reliable forecasts are achievable, which is vital in various applications, from financial markets to weather prediction. The study advocates for the continued exploration and integration of advanced deep learning models in handling complex datasets, moving beyond traditional linear approaches to embrace the whole gamut of data intricacies.

7. Conclusions

This research paper has successfully demonstrated the effectiveness of complexity-conscious prediction methods in time series forecasting. By comprehensively integrating the inherent complexities of time series data, such as nonlinearity, nonstationarity, long-term memory, asymmetry, and stochasticity, we have shown that it can achieve significantly more accurate forecasts than traditional linear models.

Our investigative focus on advanced deep learning architectures, namely Long Short-Term Memory (LSTM) networks and Generative Adversarial Networks (GANs), revealed their exceptional capability in interpreting complex, nonlinear data patterns. The comparative analysis using the Mean Absolute Percentage Error (MAPE) as a benchmark for forecasting accuracy indicated that both LSTM and GAN models perform with high precision, achieving a MAPE of less than 3.5% for all stocks in the study. Remarkably, the GAN model demonstrated outstanding performance, with a MAPE of less than 2% for all stocks in the study, thus underscoring its potent predictive capability.

These findings underscored the vital role of complexity-conscious prediction in enhancing the accuracy and reliability of time series forecasting. This method in predictive modeling recognizes and integrates the inherent complexity of data. It involves two key stages: assessing the complexity of input data and designing models tailored to this complexity. The compelling performance of LSTM and GAN models in this study challenges the traditional linear models and opens new avenues for utilizing sophisticated deep learning techniques in various practical applications. This study advocates for the broader adoption and continual refinement of such advanced methods in forecasting, emphasizing their potential to transform our understanding and prediction of complex time series data.

Ultimately, this research contributes to the evolving field of artificial intelligence by illustrating the significant benefits of embracing the full spectrum of data intricacies through complexity-conscious prediction approaches. As we continue to advance in this field, our findings highlight the importance of moving beyond conventional methodologies to leverage the rich potential of deep learning in capturing and interpreting the complex dynamics of the world around us. Though this research focused on time series data, the method of complexity-conscious prediction is relevant to any real-world use case requiring accurate predictions and informed decision-making.

Conflict of Interest

The author declares that there is no conflict of interest regarding the publication of this paper.

Author Contributions

The sole author, Raghurami Etukuru, was responsible for all aspects of this research, including conceptualization, methodology, data collection, analysis, design and settings, and writing of the manuscript. The author, Raghurami Etukuru, introduced the concept of 'Complexity-Conscious Prediction,' a novel approach that recognizes and integrates the inherent complexity of data by assessing the complexity of input data and designing models tailored to this complexity. This concept aims to significantly enhance the precision of forecasts and is particularly pertinent when data displays complex patterns and behaviors inadequately addressed by simpler predictive models.

References

[1] Etukuru, R. (2023). AI-Driven Time Series Forecasting: Complexity-Conscious Prediction and Decision-Making. iUniverse. ISBN 978-1663256713.

[2] Giles, C., & Lawrence, S. (2001). Noisy time series prediction using recurrent neural networks and grammatical inference. *Machine Learning, 44*, 161-183. https://doi.org/10.1023/A:1010884214864.

[3] Johnson, T. (2021). The influence of financial practice in developing mathematical probability. *Synthese* 198, 6291–6331. https://doi.org/10.1007/s11229-020-02636-w.

[4] Xiao, C., Xia, W., & Jiang, J. (2020). Stock price forecast based on combined model of ARI-MA-LS-SVM. *Neural Comput & Applic 32*, 5379–5388. https://doi.org/10.1007/s00521-019-04698-5.

[5] Boucher, T. (2017). Long-memory and spurious breaks in ecological experiments. *Open Journal of Statistics, 7*, 768-779. https://doi.org/10.4236/ojs.2017.75054.

[6] Dritsaki, C. (2017) An empirical evaluation in GARCH volatility modeling: Evidence from the stockholm stock exchange. *Journal of Mathematical Finance, 7*, 366-390. http://doi.org/10.4236/jmf.2017.72020.

[7] Shi, Y., & Yang, Y. (2018). Modeling high frequency data with long memory and structural change: A-HYEGARCH model. *Risks 2018*, 6, 26. https://doi.org/10.3390/risks6020026.

[8] Bhanja, S., & Das, A. (2018). Impact of data normalization on deep neural network for time series forecasting. *arXiv*. https://doi.org/10.48550/arxiv.1812.05519.

[9] Aksu, G., & Güzeller, C., & Eser, T. (2019). The effect of the normalization method used in different sample sizes on the success of artificial neural network model. *International Journal of Assessment Tools in Education*, 6, 170-192. https://doi.org/10.21449/ijate.479404.

[10] Etukuru, R. (2023). Complexity-Conscious-Prediction. GitHub. https://github.com/raghu-etukuru/Complexity-Conscious-Prediction

[11] Jordan, S., & Messner, M. (2019). The use of forecast accuracy indicators to improve planning quality: Insights from a case study. *The European accounting review, 29(2),* 337–359. https://doi.org/10.1080/09638180.2019.1577150.

[12] Vivas, E., Allende-Cid, H., & Salas, R. (2020). A systematic review of statistical and machine learning methods for electrical power forecasting with reported MAPE score. *Entropy (Basel, Switzerland),* 22(12), 1412. https://doi.org/10.3390/e22121412.