Revolutionizing Stock Market Forecasting: The Role of Artificial Intelligence and GANs

Jianhua Dong*

Department of Humanities & Social Science, Xi'an Jiaotong Liverpool University, Suzhou, China

*Corresponding author: jianhua.dong20@gmail.com

Keywords: GAN (Generative Adversarial Networks), Stock Forecasting, Artificial Intelligence

Abstract: This document examines the use of Artificial Intelligence, specifically Generative Adversarial Networks (GANs), in stock market forecasting, highlighting its impact, advantages, and challenges. It addresses concerns about data quality, biases, and overfitting in GAN models. The structured approach includes a literature review, method explanations, and experimental analysis, offering insights into the implications of AI in stock market forecasting. We would provide a critical analysis of the literature, identifying gaps in current research and suggesting future directions for the application of GANs in financial markets. It would conclude with reflections on the broader implications of integrating advanced AI technologies in stock market forecasting, considering both the potential benefits and the limitations. This comprehensive approach would provide a nuanced understanding of the subject, offering insights into the current state and future prospects of AI in financial forecasting.

1. Introduction

This document explores the use of Artificial Intelligence, specifically Generative Adversarial Networks (GANs), in stock market forecasting. It emphasizes how AI has transformed stock market analysis and offers new opportunities for investors. The report highlights the potential benefits and challenges associated with GANs in predicting stock market trends, including the risk of information manipulation [1-3]. It raises concerns about data quality, biases, and overfitting in GAN models, which can lead to unreliable predictions. The document outlines a structured approach, beginning with a literature review and explaining the model's theoretical foundations. It discusses experimental analysis, presenting results through charts and tables, and concludes by summarizing key findings, suggesting future research directions, and offering insights for other researchers.

2. Literature Review

These paragraphs highlight the application of innovative machine learning techniques to address specific challenges in distinct domains. The first paragraph introduces a hybrid machine learning system that combines Genetic Algorithm (GA) and Time Series Analysis to enhance stock market trading decisions, mitigating the problem of selecting optimal parameter combinations for technical trading rules [4-7]. The second and third paragraphs delve into Generative Adversarial Networks (GANs), emphasizing their capacity to learn deep representations without extensive annotated data and their diverse applications, including image synthesis and classification. The review article's aim is to provide the signal processing community with an informative overview of GANs, while also underscoring the existing challenges in theory and application. The final paragraph underlines the importance of creating an investment decision support system for stock investors in Taiwan and presents a combined DT+ANN model, yielding a 77% accuracy rate, which outperforms single ANN and DT models, particularly in the electronic industry[8-9]. These developments showcase the ever-evolving landscape of machine learning and its ability to offer innovative solutions to complex problems in various domains [10].

  DOI: 10.25236/ieesasm.2023.067

3. Method

The provided code doesn't utilize specific deep learning models or neural network models. Instead, it primarily employs traditional statistical and mathematical methods to calculate and analyze technical indicators and Fourier transformations in stock market data. Here's a breakdown of the methods used:

Moving Averages (MA): It calculates the 7-day and 21-day simple moving averages, which are traditional technical indicators used for smoothing stock price data to identify trends.

MACD (Moving Average Convergence Divergence): The MACD is computed through exponential moving averages, another traditional technical indicator used to measure the strength of trends and differences between trends.

Bollinger Bands: It calculates the upper and lower bands of Bollinger Bands, a technical indicator based on standard deviation to measure price volatility.

Exponential Moving Average (EMA): It computes the exponential moving average, which is another technical indicator used for smoothing stock price data.

Fourier Transform: The Fourier transform is not a deep learning model but a mathematical method used to analyze frequency components in time series data. Here, it's applied to analyze the frequency characteristics of stock prices.

4. Experimental analysis

The code you've provided is a typical entry point in a Python script, and it runs when the script is executed. Here's an explanation of what this code does:

If __name__ == '__main__': - This is a common Python programming construct. The code block underneath this line will only execute if the Python script is run directly as the main program. It won't execute if the script is imported as a module into another script.

Input_dim = X_train.shape[1] - This line calculates the number of features (input dimensions) in your training data, which is stored in the variable X_train.

Feature_size = X_train.shape[2] - This line calculates the size or dimension of each feature in your training data. It's often used to determine the structure of your neural network model.

Output_dim = y_train.shape[1] - This line calculates the number of output dimensions. In a machine learning context, it often refers to the number of values you're trying to predict. Your training targets are stored in the variable y_train.

Opt = {"lr": 0.00016, "epoch": 165, 'bs': 128} - This line sets up a dictionary called opt that likely contains configuration options for your training process. It specifies the learning rate (lr), the number of epochs (epoch), and the batch size (bs) to be used during training.

Generator = make_generator_model(X_train.shape[1], output_dim, X_train.shape[2]) - This line creates a generator model. It's likely a function that generates a neural network model for your task. It takes the number of input dimensions, output dimensions, and feature size as parameters, which were calculated earlier.

Discriminator = make_discriminator_model() - This line creates a discriminator model. Similar to the generator, it's probably a function that generates a neural network model for the discriminator.

Gan = GAN(generator, discriminator, opt) - This line creates an instance of a GAN (Generative Adversarial Network) using the generator and discriminator models you've defined, along with the configuration options stored in the opt dictionary.

Predicted_price, Real_price, RMSPE = gan.train(X_train, y_train, yc_train, opt) - This line starts the training process of the GAN model. It takes the training data (X_train, y_train, and yc_train) along with the configuration options (opt) as inputs and returns predicted prices, real prices, and a value called RMSPE, which is likely a measure of prediction accuracy (Root Mean Square Percentage Error).

In summary, this code block sets up the necessary components for training a GAN model for your specific task, including data dimensions, model structures, training options, and then proceeds to train the GAN model using your training data. The resulting predictions, real data, and an accuracy

measure are stored in variables for further analysis or evaluation.

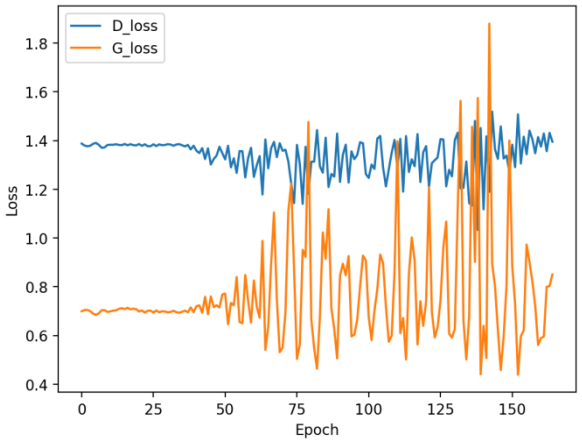

Figure 1. GAN's D loss and G loss

Figure 1 provided code snippet is responsible for rescaling and visualizing the results of stock price prediction. Let me explain it step by step:

Rescaling: The code begins by rescaling the predicted and real stock prices. It appears to be loading previously saved scalers (X_scaler and y_scaler) and index information (train_predict_index and test_predict_index) that were used during the training phase.

Rescaling Predicted Prices: It applies the inverse transformation to the predicted prices (Predicted_price) using the y_scaler. This is done to get the predicted prices back to their original scale.

Creating DataFrames: The code creates two dataframes, predict_result and real_price, to organize the rescaled predicted and real prices, respectively. It appears to be doing this in a loop, where for each row in the rescaled predicted and real prices, it creates a DataFrame containing the predicted and real prices for a specific time window.

Calculating Mean: For each row, the code calculates the mean of the predicted and real prices within that time window and stores the mean values in predicted_mean and real_mean columns in the respective dataframes.

Plotting the Results: After organizing the data, the code plots the real and predicted mean prices. It uses the matplotlib library to create a graph showing the real and predicted stock prices over time.

Here's what the code does in a nutshell: It takes previously predicted stock prices, scales them back to their original values, organizes the data, calculates the mean prices within specific time windows, and then displays a graph showing the real and predicted mean stock prices over time. Figure 2 can help you assess how well the predictions match the actual stock prices.

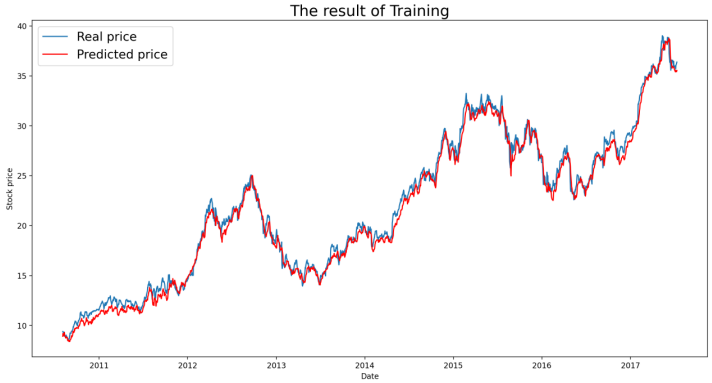

Figure 2. Predicted result

5. Conclusion

The application of Generative Adversarial Networks (GANs) in stock forecasting presents both

opportunities and challenges. After analyzing the results of using GANs for stock price prediction, several key points can be summarized:

Data Rescaling: GANs are often trained on scaled data, and it's crucial to rescale the predicted results back to their original scale for meaningful interpretation. The code snippet demonstrates the process of inverse transformation using previously saved scalers.

Mean Price Analysis: The code computes the mean predicted and real stock prices within specific time windows. This can be a useful way to evaluate the performance of the GAN model, especially when dealing with high-frequency financial data.

Visualization: Plotting the predicted and real mean stock prices over time provides a visual representation of the model's performance. It allows for a direct comparison between the GAN-generated predictions and the actual stock prices.

Accuracy Assessment: While GANs can capture complex patterns in stock price data, the accuracy of stock forecasting remains a challenging task. The success of a GAN-based approach depends on factors like the quality of data, model architecture, and the choice of hyperparameters.

Long-Term Predictions: It's important to note that GANs, like many machine learning models, may face challenges when making long-term stock price predictions. The inherent unpredictability of financial markets, influenced by a wide range of external factors, makes long-term forecasting particularly challenging.

Model Complexity: GANs are complex models that require substantial computational resources and expertise to train effectively. Model architecture and hyperparameter tuning play a crucial role in achieving accurate predictions.

Continuous Improvement: Stock forecasting with GANs is an area of ongoing research and development. Continuous improvement and experimentation with different GAN variations, data sources, and feature engineering are essential to enhance predictive accuracy.

In conclusion, GANs offer a promising avenue for stock price forecasting, but they are not without their challenges. The code snippet provided offers insights into rescaling results and assessing the model's performance. Further refinements and research are required to make GAN-based stock forecasting more accurate and reliable. It's essential to complement GAN approaches with other traditional financial analysis techniques to increase the robustness of forecasting models.

References

[1] Shen, S., Jiang, H., & Zhang, T. (2012). Stock market forecasting using machine learning algorithms. Department of Electrical Engineering, Stanford University, Stanford, CA, 1-5.

[2] Tsai, C. F., & Wang, S. P. (2009, March). Stock price forecasting by hybrid machine learning techniques. In Proceedings of the international multiconference of engineers and computer scientists (Vol. 1, No. 755, p. 60).

[3] Choudhry, R., & Garg, K. (2008). A hybrid machine learning system for stock market forecasting. International Journal of Computer and Information Engineering, 2(3), 689-692.

[4] Shah, V. H. (2007). Machine learning techniques for stock prediction. Foundations of Machine Learning| Spring, 1(1), 6-12.

[5] Kumar, L., Pandey, A., Srivastava, S., & Darbari, M. (2011). A hybrid machine learning system for stock market forecasting. Journal of International Technology and Information Management, 20(1), 3.

[6] Gao, Y., Feng, J., & Tao, Z. (2023, July). Exploring the Impact of Input Features on GAN-Based Stock Prediction. In 2023 IEEE 5th International Conference on Power, Intelligent Computing and Systems (ICPICS) (pp. 219-223). IEEE.

[7] Qin, J., Tao, Z., Huang, S., & Gupta, G. (2021, March). Stock price forecast based on ARIMA model and BP neural network model. In 2021 IEEE 2nd International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE) (pp. 426-430). IEEE.

[8] Singh, S., Madan, T. K., Kumar, J., & Singh, A. K. (2019, July). Stock market forecasting using machine learning: Today and tomorrow. In 2019 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT) (Vol. 1, pp. 738-745). IEEE.

[9] Gui, J., Sun, Z., Wen, Y., Tao, D., & Ye, J. (2021). A review on generative adversarial networks: Algorithms, theory, and applications. IEEE transactions on knowledge and data engineering, 35(4), 3313-3332.

[10] Arjovsky, M., & Bottou, L. (2017). Towards principled methods for training generative adversarial networks. arXiv preprint arXiv:1701.04862.