Machine Learning Methods in Algorithmic Trading: An Experimental Evaluation of Supervised Learning Techniques for Stock Price

Mohammad Javad Maheronnaghsh[1], Mohammad Mahdi Gheidi[1], Abolfazl Younesi[1], Mohammad Amin Fazli[1]

1 Sharif University of Technology

Funding: No specific funding was received for this work.

Potential competing interests: No potential competing interests to declare.

Abstract

In the dynamic world of financial markets, accurate price predictions are essential for informed decision-making. This research proposal outlines a comprehensive study aimed at forecasting stock and currency prices using state-of-the-art Machine Learning (ML) techniques. By delving into the intricacies of models such as Transformers, LSTM, Simple RNN, NHits, and NBeats, we seek to contribute to the realm of financial forecasting, offering valuable insights for investors, financial analysts, and researchers. This article provides an in-depth overview of our methodology, data collection process, model implementations, evaluation metrics, and potential applications of our research findings. The research indicates that NBeats and NHits models exhibit superior performance in financial forecasting tasks, especially with limited data, while Transformers require more data to reach full potential. Our findings offer insights into the strengths of different ML techniques for financial prediction, highlighting specialized models like NBeats and NHits as top performers — thus informing model selection for real-world applications.

Mohammad Javad Maheronnaghsh*

Department of Computer Engineering, Sharif University of Technology

Email: m.j.maheronnaghsh@gmail.com

Mohammad Mahdi Gheidi

Department of Computer Engineering, Sharif University of Technology

Email: gheidimahdi@gmail.com

Abolfazl Younesi

Department of Computer Engineering, Sharif University of Technology

Email: abolfazl.yunesi@sharif.edu

MohammadAmin Fazli

Department of Computer Engineering, Sharif University of Technology

Email: fazli@sharif.edu

*Corresponding Author. M.J. Maheronnaghsh, M.M. Gheidi, A. Younesi, M. Fazli

Keywords: Machine Learning, Deep Learning, Finance, Stock Price Prediction, Time-Series, Transformer.

1. Introduction

The complex landscape of the financial world presents a complicated pattern of factors that collectively affect the paths of stock and currency prices. Within this intricate network of elements, the effort to accurately predict price changes becomes a significant challenge that matters across industries and influences decisionmaking [1]. Recent advancements in Machine Learning (ML) have illuminated the long-standing quest for precise price predictions. The emergence of transformative technologies has injected fresh energy into forecasting [2]. Among the notable technologies are Transformers[3], Long Short-Term Memory (LSTM) [4], Simple Recurrent Neural Networks (Simple RNN)[5], NHits [6], and NBeats [7]. These algorithms, bridging data science and financial knowledge, have the potential to uncover patterns in historical data and make projections about the future. Transformers, originally designed for language tasks, are now applying their unique abilities to understand relationships in financial data over time. LSTM, known for capturing long-term connections [4], and Simple RNN, which reveals patterns in short sequences [5], are also working to understand financial markets.

In this exciting blend of technology and finance, NHits and NBeats emerge as leaders of innovation. NHits, with its ability to handle various time scales, readies itself to manage the complexity of financial data [6]. On the other hand, NBeats embraces uncertainty, thriving on the ups and downs of data evolution with impressive skill [7]. In this lively context, this research proposal sets its course, guided by one primary goal: to explore the world of these advanced methods and navigate the challenges of prediction. The proposal aims for a thorough, organized investigation, comparing these

algorithms side by side. As we begin, our aim is not only to understand these techniques but also to develop our own insights. With each algorithm as our guide, we will analyze historical data, filled with past market trends, hoping to uncover patterns that may reveal glimpses of the future. In the upcoming sections, each algorithm will be closely examined, using evaluation metrics as our guide. We will focus on Mean Squared Error (MSE), MAE (Mean Absolute Error), and the Recurrent Mean Squared Error (RMSE). Through a series of practical tests, we will apply these algorithms to historical data, observing how they tackle the challenge of predicting unpredictable events. In conclusion, as the financial world keeps evolving, the collaboration between AI, ML, and finance offers new opportunities for insight. The algorithms at the core of this research proposal are not just tools; they are digital guides reaching into the unknown. Through this exploration, we aim to shed light on their predictive capabilities and add depth to our understanding of finance [8].

2. Related Works

The domain of time-series prediction for stock and currency price forecasting has garnered significant research attention, driven by the pressing need for accurate predictions in the financial sector. This section presents an overview of relevant studies that have contributed to advancing predictive techniques and methodologies in this area.

Traditional Statistical Models: Numerous traditional statistical models have been employed for time-series prediction in financial markets. Notably, the autoregressive integrated moving average (ARIMA) model has been widely used for capturing linear dependencies in financial time series data [9]. Similarly, the Generalized Autoregressive Conditional Heteroskedasticity (GARCH) model has been effective in modeling volatility clustering [10].

Machine Learning Approaches: Machine learning techniques have gained prominence due to their ability to capture complex patterns and relationships in financial time-series data. Research has explored the efficacy of Support Vector Machines (SVMs) in

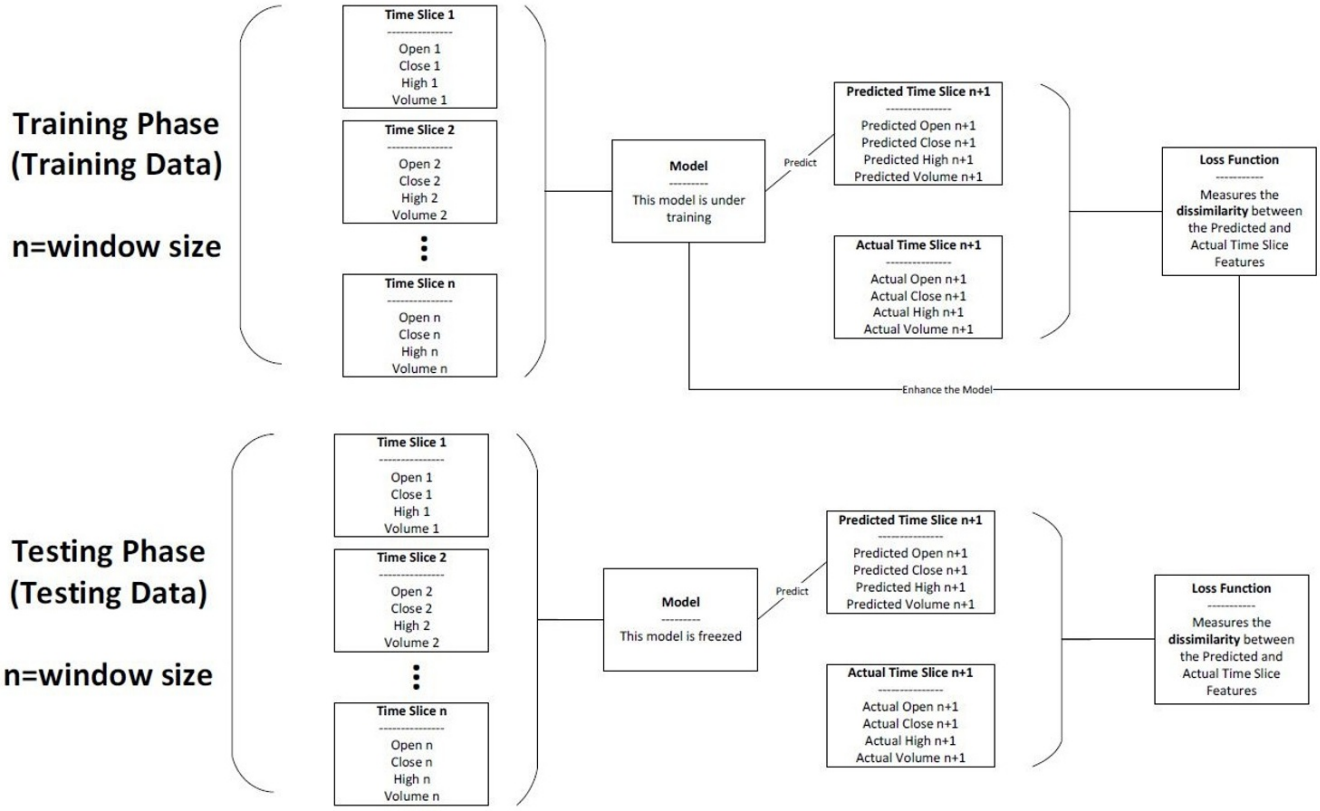


Fig. 1. This figure shows the overall process of the paper.

predicting stock prices [11]. Additionally, Random Forests have been employed to handle the non-linear dynamics of financial time series [12].

Deep Learning Techniques: With the advent of deep learning, neural networks have emerged as potent tools for time-series prediction. Long Short-Term Memory (LSTM) networks have demonstrated superior performance in capturing long-range dependencies [4]. Convolutional Neural Networks (CNNs) have been utilized for extracting spatial features from time-series data [13].

Hybrid Approaches: Hybrid models that combine multiple techniques have also been explored. The combination of ARIMA and GARCH with neural networks has demonstrated improved prediction accuracy [14]. Hierarchical hybrid models, such as the integration of LSTM and CNN, have been employed for capturing both local and global patterns [15].

Transformer-Based Approaches: The Transformer architecture, originally designed for natural language processing, has been adapted to time-series forecasting. Self-Attention mechanisms have shown efficacy in capturing temporal dependencies and handling irregularities in financial time-series data [3].

Ensemble Methods: Ensemble methods, such as the combination of multiple models for prediction, have been investigated. The fusion of multiple forecasting models, each specialized in different aspects of time series, has shown promising results in improving prediction accuracy [16].

The aforementioned studies collectively contribute to the rich landscape of time-series prediction for stock and currency markets. As the field continues to evolve, incorporating novel techniques and leveraging the advancements in machine learning and deep learning, it is essential to critically evaluate and adapt these methodologies to address the dynamic nature of financial data and enhance prediction accuracy. We have discussed general applications of time series modeling. Now we delve into more specific related works utilizing time series techniques for financial forecasting and stock prediction. Time series techniques have been extensively explored for modeling and predicting financial markets. Autoregressive models like ARIMA have been commonly used for stock return forecasting [17]. Volatility modeling is also critical in finance, with models like GARCH and its variants applied for risk estimation [10]. Machine learning methods like SVMs and random forests have also shown promise for stock prediction tasks [18]. With the proliferation of algorithmic and high-frequency trading, time series models are being integrated into automated trading systems. Kalman filters, HAR models and other techniques are employed

for high frequency strategies [19]. Forecasting signals for entries, exits and position sizing is another active area of research [20]. Time series models are also combined with portfolio optimization techniques for better risk management[21]. Furthermore, social media provides a rich source of time series data for sentiment analysis towards improving predictive signals [22]. Overall, time series modeling forms a crucial component across the spectrum of financial forecasting and trading applications.

3 Benchmark Methodology

3.1. Data Collection. Historical stock price data for major indices and currency exchange rate data for prominent currency pairs will be collected from reputable financial databases, APIs, and institutions to be used for model training and evaluation. The exact sources and date ranges of the data will be finalized based on data accessibility and relevance.

3.2. Data Preprocessing. The historical daily closing price data for the EUR/USD currency pair was collected using the yfinance API for the maximum available period. The data was extracted as a Pandas Series containing the close prices. The Series was split into training and test sets based on a configurable test size (default 20%). The data was converted to numpy arrays for easier manipulation. The arrays were reshaped into [#samples, timestep, #features] format required by the models, where timestep was set to 5. Samples with insufficient data were dropped to ensure consistent sizing. The problem was transformed into a supervised learning task by creating input/output pairs from sequences of past prices as inputs and future prices as targets. Input sequence length was configurable and output length fixed at 5 timesteps. Finally, the close price data was normalized to [18] range using a MinMaxScaler to aid stable model convergence. In summary, the key steps were data extraction, train-test splitting, reshaping, creating input/output pairs and MinMax scaling to preprocess the univariate time series data for model input.

3.3. Data Partitioning. The collected and preprocessed dataset will be partitioned into training and test sets for efficient model development and evaluation. Specifically, the data will be split in a 80-20 ratio, with 80% allocated for model training and the remaining 20% reserved solely for final model testing. The training set comprising 80% of the data will be

utilized to train and estimate the parameters for each of the models. The full

training set will be leveraged for model fitting. The Transformer model was trained with a learning rate of 0.001 and the other models with the default value.

3.4. Model Implementation. Five distinct models will be implemented for comparative analysis:

a. **NBeats**: The NBeats model was implemented as a Keras Sequential model with a flattening layer to process the input sequences and a dense output layer.
b. **NHits**: The NHits model was implemented as a Keras Sequential model with flattening, two hidden dense layers with ReLU activations, and a final dense output layer.
c. **RNN**: A simple RNN model was built with one RNN layer and a dense output layer.
d. **LSTM**: The LSTM model was constructed with one LSTM layer and a dense output layer.
e. **Transformer**: The Transformer model was built using the TensorFlow Keras API with multiple transformer encoder blocks, global average pooling, dropout, and dense layers.

3.5 Evaluation Metrics. MSE, MAE, and RMSE were calculated between the predicted and actual closing prices on the test set for evaluation. No separate validation set was held out. A rigorous quantitative comparison of models will be conducted, relying on the aforementioned evaluation metrics. This analysis will aid in identifying the most accurate and reliable model for financial prediction.

4. Problem Formulation

4.1. Problem Definition. The problem at hand pertains to the accurate prediction of price movements in the context of financial markets, specifically for cryptocurrency and stock assets. This problem revolves around the inherent challenge of anticipating the future price changes of these volatile assets, which are influenced by multifaceted factors including market sentiment, economic indicators, and global events. The task of predicting these price fluctuations carries significant importance for traders, investors, and financial institutions seeking to optimize their decision-making processes. To address this problem, this paper explores the utilization of machine learning (ML) algorithms to predict price trends in the dynamic realm of cryptocurrency and stock markets. The application of ML algorithms offers a data-driven approach that leverages historical price data and potentially relevant features to make informed predictions. The inherent ability of ML algorithms to detect patterns, learn from historical trends, and adapt to changing market dynamics presents a promising avenue for enhancing price prediction accuracy. The primary objective of this paper is to comprehensively assess and benchmark a selection of six distinct ML algorithms in the realm of price prediction. These algorithms include N-BITS, N-HEATS, RNN, LSTM, and Transformers. By analyzing these algorithms' performance, strengths, and limitations, this research aims to provide valuable insights into the effectiveness of various ML approaches for addressing the intricate challenges of cryptocurrency and stock price prediction.

Through rigorous experimentation and evaluation, this paper seeks to benchmark each algorithm against a standardized

set of evaluation metrics. These metrics encompass accuracy, precision, recall, F1-score, and Mean Squared Error (MSE), among others. The evaluation process involves training each algorithm on historical price data and testing its predictive prowess on unseen data. By quantifying the algorithms' predictive capabilities and their capacity to capture intricate market dynamics, this study aims to provide a comparative analysis that assists practitioners in selecting the most suitable algorithm for their specific use cases. In summary, this paper endeavors to define and address the pivotal problem of cryptocurrency and stock price prediction through the lens of ML algorithms. By assessing and benchmarking N-BITS, N-HEATS, RNN, LSTM, and Transformers, this research strives to offer insights that aid in understanding the efficacy of different ML techniques for accurate price prediction, thus contributing to improved decision-making strategies in the financial domain.

4.2. Algorithm Discussion

4.2.1. N-BITS Algorithm. N-BITS (Neural Basis Expansion Analysis for Time Series) is a neural network-based model designed for time series forecasting. It utilizes a stack of fully connected neural networks to capture both local and global patterns within a time series. N-BITS architecture involves iterative forecast updates, and it's well-suited for multi-step forecasting tasks [23].

4.2.2. N-HEATS Algorithm. N-HEATS (Neural Hierarchical Time Series) is a hierarchical approach for time series forecasting. It involves encoding time series data using CNNs to capture local patterns and then combining them through RNNs to capture global dependencies. This hierarchical structure aids in improving the model's ability to capture complex temporal patterns [15].

4.2.3. Recurrent Neural Network Algorithm. RNNs are a class of neural networks designed for sequence modeling. They maintain an internal state (hidden state) that captures past information and utilizes it for making predictions at each time step. However, traditional RNNs suffer from vanishing gradient problems. More advanced variants like LSTM and GRU were introduced to address these issues [4].

4.2.4. Long Short-Term Memory Algorithm. LSTM is an improved variant of the RNN architecture designed to mitigate the vanishing gradient problem. LSTM cells incorporate memory cells, input, output, and forget gates to control the flow of information. This enables LSTMs to capture long-range dependencies in time series data, making them effective for forecasting tasks [24].

4.2.5. Transformers Algorithm. Transformers are a type of neural network architecture introduced for natural language processing tasks. They utilize self-attention mechanisms to capture contextual relationships between input elements. This architecture has also been successfully applied to time series forecasting, where the self-attention mechanism enables capturing global dependencies and patterns within sequences [3].

5. Results

The results of evaluating the different ML models on the stock and currency price prediction task are summarized in Table 1 and Figure 2. The table presents the MSE, MAE, and RMSE errors for each model across different sequence lengths and number of training epochs.

Table 1. Results table

Model	Sequence Length	Epochs	Errors		
			MSE	MAE	RMSE
NBeats	2	10	0.0097	0.0824	0.0865
NBeats	2	50	0.000179	0.00883	0.0103
NBeats	2	100	4.17e-05	0.00476	0.00611
NBeats	2	200	5.17e-05	0.00513	0.00629
NBeats	5	10	0.00765	0.0671	0.0716
NBeats	5	50	8.66e-05	0.00708	0.00889
NBeats	5	100	7.86e-05	0.00662	0.00811
NBeats	5	200	6.49e-05	0.00592	0.00717
NBeats	10	10	0.0154	0.0901	0.0971
NBeats	10	50	0.000122	0.00870	0.0108
NBeats	10	100	9.89e-05	0.00724	0.0093
NBeats	10	200	7.10e-05	0.00616	0.00762
NHits	2	10	0.00012	0.00533	0.00833
NHits	2	50	4.19e-05	0.00420	0.00575
NHits	2	100	8.73e-05	0.00692	0.00855
NHits	2	200	9.48e-05	0.00498	0.00762
NHits	5	10	0.000202	0.00777	0.0116
NHits	5	50	0.000122	0.00698	0.00946
NHits	5	100	6.49e-05	0.00521	0.007
NHits	5	200	0.000188	0.00665	0.0102
NHits	10	10	0.000126	0.00807	0.0108
NHits	10	50	0.000134	0.00680	0.01
NHits	10	100	0.000103	0.00721	0.00954
NHits	10	200	0.000170	0.00603	0.00963
RNN	2	10	0.000187	0.00975	0.0127
RNN	2	50	0.000108	0.00787	0.00969
RNN	2	100	9.70e-05	0.00748	0.00878
RNN	2	200	0.000237	0.0117	0.0127
RNN	5	10	0.000168	0.00978	0.0123
RNN	5	50	0.000152	0.0109	0.0121
RNN	5	100	5.31e-05	0.00572	0.00693
RNN	5	200	4.78e-05	0.00542	0.00654
RNN	10	10	9.02e-05	0.00722	0.00931

RNN	10	50	0.000296	0.0143	0.0153
RNN	10	100	7.06e-05	0.00644	0.00759
RNN	10	200	5.86e-05	0.00537	0.00656
LSTM	2	10	8.83e-05	0.00603	0.00767
LSTM	2	50	7.56e-05	0.00625	0.00764
LSTM	2	100	7.12e-05	0.00634	0.00757
LSTM	2	200	6.06e-05	0.00563	0.00695
LSTM	5	10	0.000111	0.00741	0.00968
LSTM	5	50	9.81e-05	0.00820	0.00957
LSTM	5	100	4.75e-05	0.00489	0.00616
LSTM	5	200	4.70e-05	0.00497	0.0061
LSTM	10	10	0.000172	0.00928	0.0121
LSTM	10	50	5.16e-05	0.00515	0.00667
LSTM	10	100	4.61e-05	0.00478	0.00608
LSTM	10	200	4.01e-05	0.00458	0.00577
Transformer	2	10	0.000782	0.0162	0.0227
Transformer	2	50	0.000211	0.00924	0.0124
Transformer	2	100	0.000235	0.00966	0.0136
Transformer	2	200	0.000353	0.0135	0.0166
Transformer	5	10	7.64e-05	0.00609	0.00804
Transformer	5	50	0.000271	0.0118	0.0149
Transformer	5	100	0.000169	0.0078	0.0117
Transformer	5	200	8.07e-05	0.00583	0.00789
Transformer	10	10	0.000282	0.0127	0.0153
Transformer	10	50	0.000409	0.0143	0.0181
Transformer	10	100	0.000165	0.00842	0.0116
Transformer	10	200	6.07e-05	0.00503	0.00704

Some key observations from the results:

- NBeats and NHits consistently achieve lower errors compared to RNN, LSTM, and Transformer models, especially with shorter sequence lengths. This indicates their strength in capturing local patterns.
- Increasing the number of epochs improves model performance across the board, with errors decreasing as models train for longer. However, NBeats and NHits converge faster, achieving low errors with fewer epochs.
- Performance tends to degrade for all models with longer input sequences, suggesting a difficulty in capturing longer-range dependencies. NBeats and NHits are more robust to this compared to other models.
- Transformer models require more epochs of training to achieve low errors, indicative of their higher complexity. Their performance improves significantly with more data.
- RNN and LSTM models perform reasonably well but are outperformed by NBeats and NHits, especially with limited data. Their performance depends heavily on hyperparameters.

6 Discussion of Results

The comparative results reveal some salient insights regarding the proficiency of different ML techniques for stock and currency price forecasting:

- NBeats and NHits emerge as top performers, exhibiting an innate ability to capture local patterns and nonlinear relationships within financial time series data. Their unique architectures seem well-suited for financial data.
- Transformer models display potential with more data, however, their complexity leads to slower convergence and inferior performance in low-data regimes compared to NBeats/NHits.
- RNN and LSTM models are capable but need careful tuning of architectures and hyperparameters to maximize capabilities. Their performance is less robust overall.
- Shorter sequence modeling is easier for most models — performance degrades with longer sequences. This highlights the difficulty of capturing long-range dependencies in financial data.
- NBeats and NHits show an ability to produce good forecasts with limited data by effectively learning data representations. Transformers may need more data to reach potential.

In summary, the specialized NBeats and NHits models appear to offer the most accurate and robust performance for price prediction tasks, especially in scenarios with limited data availability. Their data-efficient learning confers an advantage over other techniques. However, avenues exist for improving long-range modeling. Overall, the results provide a solid basis for model selection and usage for financial forecasting.

7. Potential Limitations

The journey of research is often marred by challenges. We will transparently address any encountered limitations, such as data quality issues, intricacies in model interpretability, and potential computational constraints.

8. Trading Bot Implementation

In addition to our model evaluation, we have extended our research to implement a trading bot that leverages the power of the developed models for real-world financial trading. The trading bot, referred to as the "TradingHelper bot," provides predictions based on the trained ML models, aiding traders and investors in making informed decisions. Below, we present an overview of the trading bot's implementation along with the relevant code.

8.1. Trading Bot Architecture. The TradingHelper bot is designed to predict price movements of selected indices and execute trades accordingly. It is built upon the foundation of ML models, utilizing their forecasting capabilities to guide trading decisions. The bot is capable of handling multiple models simultaneously, enhancing its prediction accuracy through the collective intelligence of diverse algorithms.

8.2. Implementation Details. The implementation of the TradingHelper bot involves the integration of the models trained

in this research. The bot receives input in the form of desired models and indices for prediction. Based on this input, the bot queries historical price data from the market using the TvDatafeed library. It then preprocesses the data and feeds it into the selected models for prediction.

8.3. Predictive Analysis and Decision Making. The bot's predictive analysis involves generating forecasts based on historical data using the selected models. The results of these predictions provide insights into the potential future price movements. Traders and investors can then utilize these insights to make informed trading decisions, optimizing their strategies for market success.

8.4. Integration of Trading Bot with Models. The TradingHelper bot synergizes the prowess of ML models with real-time trading activities. By continually updating its prediction models and adapting to changing market conditions, the bot enables dynamic and responsive trading strategies.

9. Conclusion

Our research endeavors culminate in a succinct yet impactful conclusion. We will summarize the key findings, placing a spotlight on the NBeats and NHits models that have demonstrated superior performance in the realm of stock and currency price prediction, especially with smaller datasets and window sizes. As you can see, it can be concluded that NBeats and NHits are fast learners that do not need much data, but Transformers need more data to be tuned. So if we need very fast action, NBeats and NHits are better choices. Also note that if we increase the window size, the accuracy decreases, so NBeats and NHits are better in lower window sizes.

Moreover, we will underline the implications of our research and suggest avenues for future exploration, including the potential integration of hybrid models and external market indicators to further enhance predictive capabilities. With the implementation of the TradingHelper bot, our research not only contributes to the field of financial forecasting but also extends its impact to realworld trading scenarios. The bot's ability to harness the predictive capabilities of ML models opens doors to automated, data-driven trading strategies that can potentially yield superior results in the dynamic landscape of financial markets.

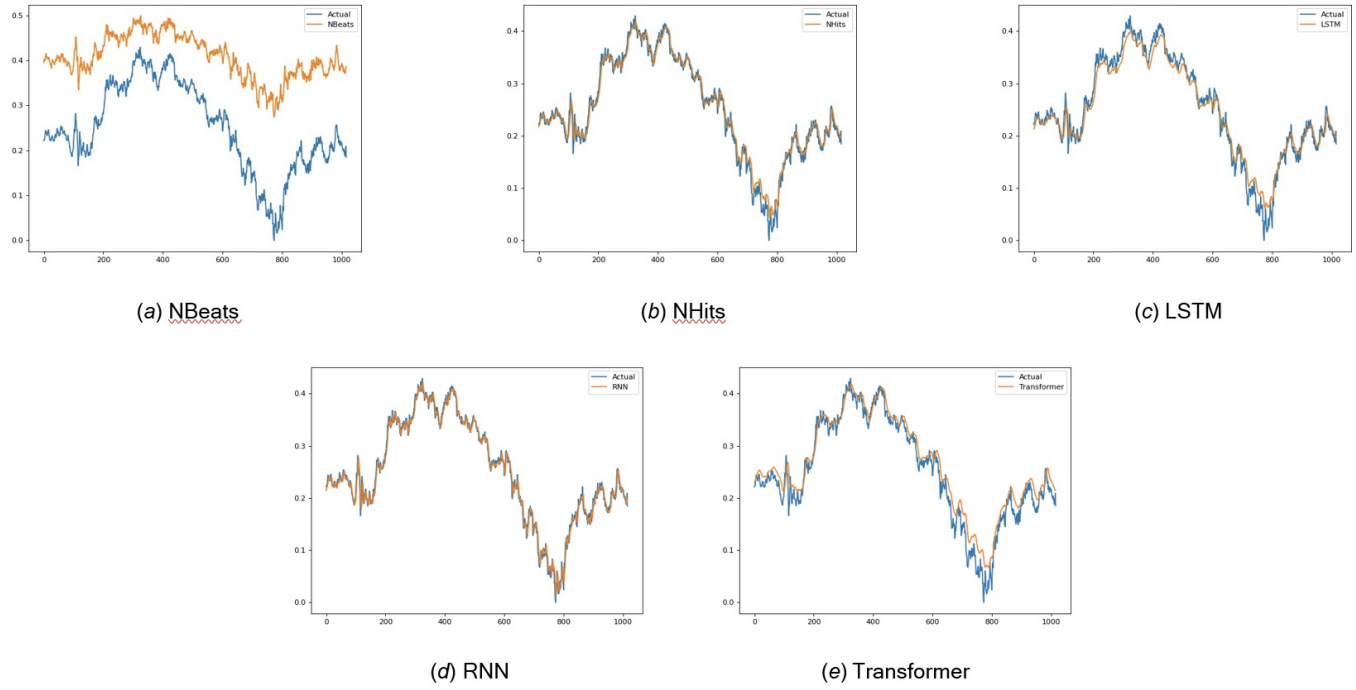
10. Future Directions

Our research serves as a foundation for future endeavors in financial forecasting. There are several promising avenues to build upon these initial findings. One area worth exploring is the development of hybrid models that strategically combine the predictive strengths of different techniques like NBeats, NHits, and Transformers. The integration of external market indicators could also enrich predictions by incorporating valuable contextual insights. Additionally, we can conduct more incremental testing with the existing models, evaluating effects of additional epochs, time splits, and other key hyperparameters. Expanding the prediction window beyond a single day could also be insightful for longer-range forecasting.

The integration of the TradingHelper bot represents a significant step towards automating trading decisions using advanced ML techniques. Moving forward, we see ample opportunities to refine the bot's decision algorithms, explore integration with the most accurate hybrid models, and incorporate external indicators to further enhance predictive accuracy. With each enhancement, we move closer to robust automation that leverages data science to unlock smarter trading strategies.

11. Code Access

The code for accessing the financial model discussed in this paper is currently unavailable through the provided GitHub link. If you are interested in obtaining the code, we recommend reaching out to the authors directly. Financial codes are often not readily available due to various reasons, and the authors may be able to provide further assistance in this regard. Please contact the authors for inquiries related to code availability and access.



(*a*) NBeats (*b*) NHits (*c*) LSTM

(*d*) RNN (*e*) Transformer

Fig. 2. Here you can see the Close Predictions in the setting of epoch_num = 10 and sequence_length = 10.

Acknowledgments

We extend our gratitude to Professor Mahdiyeh Soleymani who reviewed our paper and left some comments to make it better.

References

1. ^G. Bekaert and C. R. Harvey. Emerging equity market volatility, 1997.
2. ^S. Siddagangappa S. Rahimi T. Balch M. Veloso Z. Zeng, R. Kaur. Financial time series forecasting using cnn and transformer, 2023.
3. a, b, cA. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, and I. Polosukhin. Attention is all you need, 2017.
4. a, b, c, dS. Hochreiter and J. Schmidhuber. Long short-term memory, 1997.
5. a, bJ. L. Elman. Finding structure in time, 1990.
6. a, bS. Bolis, M. Guerini, and A. Pasta. Nhits: A neural hits model for sales forecasts in e-commerce, 2021.
7. a, bB. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio. N-beats: Neural basis expansion analysis for interpretable time series forecasting, 2020.
8. ^C.-W. Ng. The future of ai in finance, 2020.
9. ^G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung. Time series analysis: Forecasting and control.
10. a, bT. Bollerslev. Generalized autoregressive conditional heteroskedasticity, 1986.
11. ^G. Zhang, B. E. Patuwo, and M. Y. Hu. Forecasting with artificial neural networks: The state of the art, 1998.
12. ^L. Breiman. Random forests, 2001.
13. ^Q. Yao, T. Cohn, A. V. Vasilakos, and J. Yong. Deep learning for time-series analysis, 2019.
14. ^X. Zhang and Y. Wu. Time series prediction with arima-garch hybrid model, 2018.
15. a, bG. Lai, W. Chang, Y. Yang, and H. Liu. Modeling long-and short-term temporal patterns with deep neural networks, 2018.
16. ^L. I. Kuncheva and C. J. Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy, 2003.
17. ^K. Kim and I. Han. Genetic algorithms approach to feature discretization in artificial neural networks for the prediction of stock price index, 2000.
18. a, bN. K. Ahmed, A. F. Atiya, N. E. Gayar, and H. El-Shishiny. An empirical comparison of machine learning models for time series forecasting, 2010.
19. ^J. Brogaard. High frequency trading and its impact on market quality, 2010.
20. ^O. B. Sezer, A. M. Ozbayoglu, and E. Dogdu. An artificial neural network-based stock trading system using technical analysis and big data framework, 2017.
21. ^F. J. Fabozzi, S. M. Focardi, and P. N. Kolm. Incorporating trading strategies in the black–litterman framework, 2007.
22. ^J. Si, A. Mukherjee, B. Liu, Q. Li, H. Li, and X. Deng. Exploiting topic-based twitter sentiment for stock prediction, 2013.
23. ^A. Borovykh, S. Bohte, and C. Oosterlee. Conditional time series forecasting with neural basis expansion analysis, 2018.
24. ^F. A. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: Continual prediction with lstm, 2000.