Evaluation of Stock Market Prediction Techniques

Abhishek Malaviya
School of Computing
Dublin City University
abhishek.malaviya2@mail.dcu

Dr. Andrew McCarren
Head of School of Computing
Dublin City University
andrew.mccarren@dcu.ie

***Abstract*—This study evaluates the performance of traditional and deep learning models in predicting stock prices using a mix of financial and behavioral data. The Dow Jones Industrial Average (DJIA) was forecasted for one-day, three-day, and seven-day horizons using ARIMA, LSTM, and Temporal Convolutional Networks (TCN). The input consisted of 50+ feature engineered signals like technical indicators with sentiment and intent signals from news and Reddit data.**

Results showed that ARIMA performed best for short-term forecasts, while the stacked LSTM gave the most reliable performance across all time frames. TCN performed best compared to other models across multiple time horizons. The findings highlight the importance of using multimodal data and selecting models based on forecast horizon.

***Keywords*– Stock prediction, ARIMA, LSTM, TCN, Sentiment analysis, Multimodal data, Time series forecasting.**

I. Introduction

The global stock market, valued at over $100 trillion, plays a crucial role in shaping investment flows, corporate performance, and broader economic conditions [1]. Despite technological advancements, predicting future stock prices remains highly complex due to the dynamic, volatile, and nonlinear nature of financial time series data [2], [3]. Prices are influenced not only by historical patterns and technical indicators but also by external factors such as news sentiment, social media activity, and investor behavior [4]–[6].

Traditional methods such as Autoregressive Integrated Moving Average (ARIMA) have been used extensively for financial forecasting [7], offering reasonable short-term performance under stationary assumptions. However, their inability to model nonlinear dependencies limits their effectiveness in capturing market volatility [1]. To overcome these limitations, deep learning models such as Long Short-Term Memory (LSTM) networks and Temporal Convolutional Networks (TCNs) have been proposed [8]–[10]. LSTM models are designed to retain long-range dependencies in time-series data, while TCNs leverage parallelized convolutions and causal padding to learn complex temporal features effectively [10], [11].

In recent studies, hybrid and multimodal approaches have been explored to enhance model robustness and accuracy. These include combining structured time series with unstructured data sources like news headlines and social sentiment [4], [5], [12]. Notably, multimodal systems have demonstrated strong potential in capturing behavioral signals that precede market trends [13], [14].

This research aims to compare the performance of ARIMA, LSTM, and TCN models in forecasting the closing price of the Dow Jones Industrial Average (DJIA) index across multiple forecast horizons—one day, three days, and seven days ahead. A multimodal dataset was constructed using historical stock data with sentiment and intent signals extracted from financial news and Reddit discussions. Evaluation was carried out using standard regression metrics including Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and the Coefficient of Determination (R^2). The results show that TCN outperforms both ARIMA and LSTM across most timeframes, except for the longest horizon (t+7), where ARIMA showed marginally better generalization under volatility [15], [16].

II. Related Work

lit review goes here

III. Dataset and Feature Engineering

This study uses a feature-engineered multimodal dataset focused on the Dow Jones Industrial Average (DJIA), spanning over 1900 trading days (~8 Years) was choosen from Kaggle [17]. There were three datasets - financial data, news articles, and Reddit discussions, captures both quantitative and behavioral signals that may influence market movements.

A. Technical Indicators

To represent core market movements, I included three key technical features: These features were engineered from historical price data[1]:

- **Log Returns**: Captures the daily log return of the closing price to measure relative change in price.

$$R_t = \log\left(\frac{P_t}{P_{t-1}}\right)$$

  where R_t is the log return on day t, and P_t is the closing price on day t.

[1]LaTeX formatting for formulas were assisted by ChatGPT [18].

- **Volatility log10**: Represents the 10-day rolling standard deviation of log returns, scaled by the square root of 252 to annualize the volatility.

$$\sigma_t = \sqrt{252} \times \text{StdDev}(R_{t-9}, R_{t-8}, \ldots, R_t)$$

where σ_t is the estimated annualized volatility at time t.
- **Percentage Change**: Computes the simple daily percentage change in price.

$$\Delta_t = \left(\frac{P_t - P_{t-1}}{P_{t-1}} \right) \times 100$$

where Δ_t is the percent change at time t.

B. Sentiment and Intent Features

Behavioral features were computed using Natural Language Processing (NLP) techniques applied to both news headlines and Reddit posts. Three independent sentiment scoring techniques were used [4]:

- **VADER sentiment** for analyzing social media text.
- **FinBERT sentiment**—a transformer model fine-tuned on financial documents.
- **Smart sentiment**, a technique that corrected FinBERT outputs based on hand-written financial trigger words (e.g., down-weighting neutral sentiment if keywords like "crash" or "crisis" appeared).

To contextualize sentiment with behavioral intent, I implemented a custom keyword-matching dictionary to extract psychological signals from both Reddit and news text. The model computed nine intent types: *buying, selling, uncertainty, urgency, prediction, fear, greed, question*, and *action*.

Each intent feature was calculated as[2]:

$$I_i = I_i^{\text{news}} + I_i^{\text{reddit}},\ i \in \{\text{buying, selling, fear, uncertainty}, \ldots\}$$

However, the raw intent scores ranged from -2 to $+1.8$, while sentiment scores (e.g., VADER, FinBERT) lie within $[-1, +1]$. To align these distributions and avoid introducing scale bias during model training, all intent values were **min-max normalized** to the $[-1, 1]$ range.

Finally, sentiment and intent features were fused to create hybrid behavioral indicators:

$$\begin{aligned} \text{sentiment\_minus\_fear} &= \text{finbert\_final\_sentiment} \\ &\quad - \text{normalized\_fear\_intent} \\ \text{sentiment\_minus\_uncertainty} &= \text{finbert\_final\_sentiment} \\ &\quad - \text{normalized\_uncertainty\_intent} \end{aligned}$$

These feature-engineered intents aimed to identify emotional nature of the stock market, such as positive sentiment overshadowed by fear or uncertainty—critical signals. All features were scaled and merged into 'multimodal_dataset' for time series modeling.

[2]LaTeX formatting for formulas were assisted by ChatGPT [18].

C. Target Variables

The forecasting task is framed as a **multi-output regression problem**, where the models predict three future target variables:

- **t+1** — Closing price of the next trading day,
- **t+3** — Closing price three days ahead,
- **t+7** — Closing price seven days ahead.

This enables the models to learn both short-term and mid-term temporal dependencies in the architecture. All target variables were normalized using `MinMaxScaler`, and the input features were segmented into sliding window of **60 time steps**(for LSTM) and window of **30 time steps** for TCN, which leverages dilated convolutions and does not require long historical sequences.

IV. Methodology

This study compares three models—ARIMA, LSTM, and Temporal Convolutional Network (TCN)—for forecasting DJIA closing prices at three horizons: t+1, t+3, and t+7. A multimodal dataset of technical, sentiment, and intent features was used. Supervised learning sequences were constructed using a sliding window approach over the time series.

A. Modeling Pipeline

The pipeline followed five key steps:

1) **Data Preprocessing:** Missing values were dropped because during feature engineering percentage change and volatilty log 10 introduced NaN values which only could be dropped. All sentiment and intent features were scaled to the range $[-1, 1]$ using `MinMaxScaler`.
2) **Sequence Generation:** For LSTM, 60-step sequences (3 months) were used. For TCN, a 30-step window was found optimal after experimentation.
3) **Train-Test Split:** A manual 80:20 chronological split was applied to maintain temporal order and prevent lookahead bias. The model was trained only on train dataset in a separate notebook and same for the test.
4) **Model Training:** Each model was reinitialized for every forecast horizon to avoid memory leakage. TensorFlow memory was explicitly cleared between runs to ensure accurate and isolated training.
5) **Evaluation:** Performance was measured using RMSE, MAE, and R^2 on training and unseen data. Additionally, residual and forecast plots were generated for visual analysis.

B. Baseline: ARIMA

ARIMA was implemented as a univariate baseline model. For each forecast horizon, the model was retrained at each time step using past values up to t. The auto arima function was used to automatically determine the best parameters $(p, d, q) = (1, 1, 1)$, and predictions were made in a rolling window fashion. Model performance was evaluated on a true future holdout series, not just in-sample forecasts. After each training cycle, and models were serialized with joblib for testing on the unseen data.

C. LSTM Architectures

Two variants of LSTM were trained for each forecast horizon using a supervised learning setup:

- **Single-layer LSTM:** This model used 64 hidden units followed by a dropout layer and a dense output unit. It served as a fast and lightweight benchmark.
- **Stacked LSTM:** A deeper model with two LSTM layers (64 units + 32 units), both regularized using `L2` constraints. Dropout was used after each layer to improve generalization. This version was found to generalize better across short-term and mid-term horizons.

Input sequences of shape (50, features) were used. The model predicted one output value for each sample, corresponding to the price at $t + h$. Targets were normalized independently, and inverse scaling was performed post-prediction.

All training runs used the Adam optimizer with a learning rate of 1×10^{-4}, mean squared error (MSE) loss, batch size 32, and early stopping on validation loss with a patience of 5 epochs. The models and scalers were saved for reuse during testing.

D. Temporal Convolutional Network (TCN)

The TCN model was trained to predict the **log return** of DJIA prices at future horizons. The predicted log return was then used to reconstruct the actual price using:

$$\hat{P}_{t+h} = P_t \cdot e^{\hat{r}_{t+h}}$$

where $\hat{r}_{t+h}$ is the predicted log return and P_t is the closing price at the end of the input window.

TCN architecture:

- **Input:** Sequences of 30 time steps and all available features.
- **Architecture:** 1D dilated causal convolutions with 32 filters and kernel size 2, followed by `GroupNormalization`, dropout (0.2), and a dense output layer.
- **Training:** Adam optimizer, Huber loss, batch size 32, early stopping with patience 5.
- **Clipping:** Predicted log returns were clipped between $[-0.15, 0.15]$ to avoid exponential blow-ups during price reconstruction.

As with LSTM, the TensorFlow graph was cleared between runs. Models were saved along with their respective scalers and feature metadata.

E. SHAP Analysis for TCN

To interpret the contribution of individual features across different forecast horizons, SHAP (SHapley Additive exPlanations) analysis was performed on trained TCN models using the `KernelExplainer`.

Key observations:

- For **t+1**, the most important signals were `Smart reddit sentiment`, `vader reddit sentiment`, and `total greed intent`.

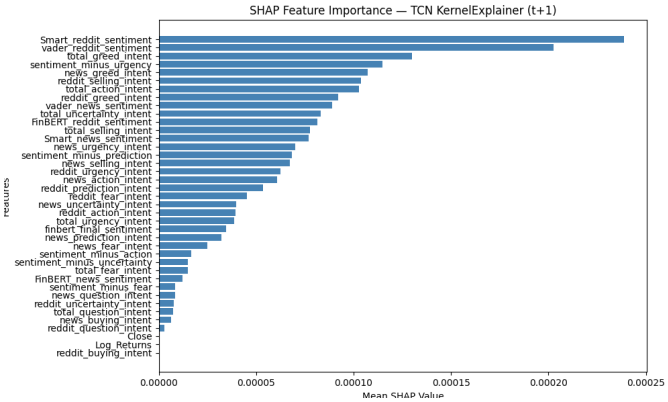

Fig. 1. SHAP feature importance — TCN prediction at t+1 horizon

- For **t+3**, features like `reddit prediction intent`, `news urgency intent`, and `news greed intent` dominated.
- For **t+7**, higher-level sentiment aggregates such as `Smart reddit sentiment` and `Smart news sentiment` had the most influence.

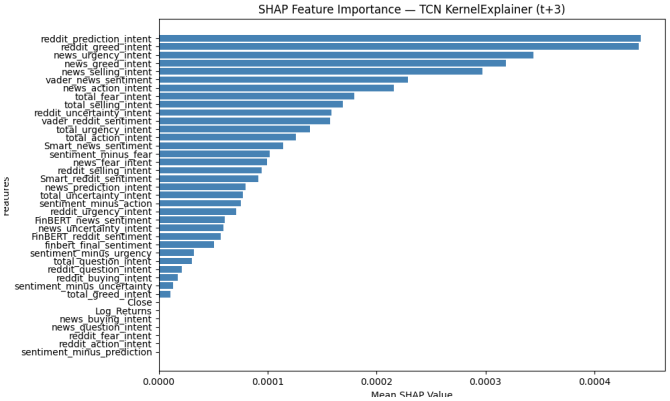

Fig. 2. SHAP feature importance — TCN prediction at t+3 horizon

Interestingly, traditional technical indicators such as `Close` and `Log Returns` ranked among the least influential. This validated the hypothesis that multimodal behavioral signals are more informative for modern financial forecasting than legacy technical metrics.

TCN was robust in learning the patterns between feature engineered behavioural intent and the actual price of the stock.

After analyzing SHAP output, there is a pattern which TCN did learn- Stock market is heavily influenced primarily on traders behavior compared to other technical indicators and metrics. This setting simulates real-life decisions that traders make while purchasing a stock.

Implementing SHAP really helped fine-tune TCN and reducing the feature set as during training, TCN was only

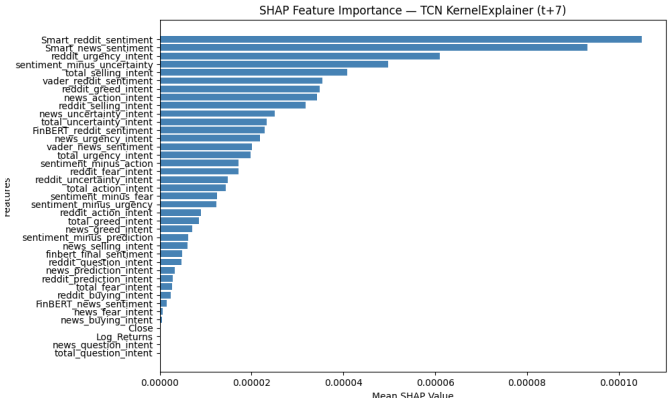

Fig. 3. SHAP feature importance — TCN prediction at t+7 horizon

able to learn noise which was indicating that the model was overfitting.

V. Results and Evaluation

This section presents the performance of all models—ARIMA, LSTM (Simple and Stacked), and TCN—on forecasting the DJIA closing price at future horizons of t+1, t+3, and t+7. Evaluation metrics include Coefficient of Determination (R^2), Root Mean Square Error (RMSE), and Mean Absolute Error (MAE). All models were trained on 80% of the data and evaluated on the remaining 20%, with TensorFlow memory cleared between each model run to ensure isolated performance measurement.

A. ARIMA Performance

TABLE I
ARIMA Train and Test Performance

Horizon	Train			Test		
	R^2	RMSE	MAE	R^2	RMSE	MAE
t+1	0.9921	238.87	136.61	**0.8655**	**228.79**	**170.45**
t+3	0.9877	297.54	190.21	0.7433	316.38	236.20
t+7	0.9800	380.06	260.24	0.5662	413.11	317.17

The ARIMA model showed excellent performance for short-term forecasting (t+1), with the highest test R^2 among all models in that category. This aligns with its strength in capturing autoregressive and trend components in stationary series. However, the model's performance dropped off rapidly for longer horizons due to its inability to incorporate external or nonlinear signals like sentiment or volatility.

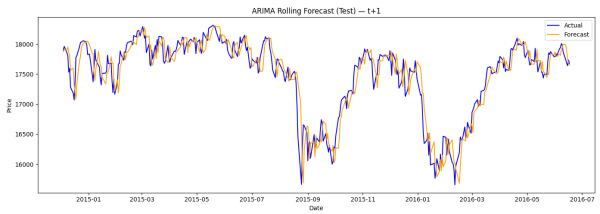

Fig. 4. ARIMA Forecast — t+1 Horizon

The t+1 prediction plot reveals that ARIMA closely follows the actual DJIA prices, particularly during low-volatility phases. Minor lags appear during sharp directional changes, suggesting the model's lag in adapting to market shocks.

B. LSTM (Simple)

TABLE II
Simple LSTM Train and Test Performance

Horizon	Train			Test		
	R^2	RMSE	MAE	R^2	RMSE	MAE
t+1	0.9881	294.14	225.36	**0.7606**	**319.58**	**263.98**
t+3	0.9863	315.44	241.68	0.2669	559.01	459.02
t+7	0.9855	323.81	246.69	0.4156	501.07	384.07

The simple LSTM architecture generalizes really well on the t+1 horizon, with a test R^2 of 0.76. However, generalization quickly deteriorates for longer-term targets, reflecting underfitting and the limited expressiveness of a shallow network when handling multimodal volatility-prone financial data.

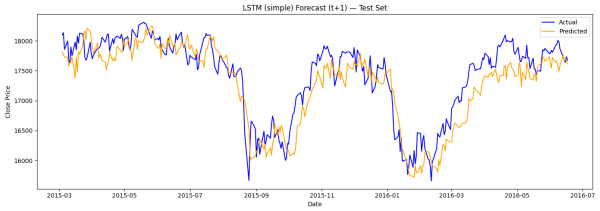

Fig. 5. Simple LSTM Forecast — t+1 Horizon

The t+1 plot demonstrates that while the model captures the directional movement of the price trend, it shows delay around peaks and valleys, suggesting insufficient sensitivity to sudden changes despite using sentiment-enhanced features.

C. LSTM (Stacked)

TABLE III
Stacked LSTM Train and Test Performance

Horizon	Train			Test		
	R^2	RMSE	MAE	R^2	RMSE	MAE
t+1	0.9826	355.83	272.09	0.3267	535.94	454.12
t+3	0.9858	320.75	251.53	0.3985	506.37	409.86
t+7	0.9700	466.63	365.38	**0.4252**	**496.95**	**376.23**

The stacked LSTM model overfit slightly on the training set, especially at t+1 where test R^2 dropped significantly. However, its deeper structure helped stabilize predictions at t+7, where it beats the simple LSTM model by a small margin. This highlights its utility in longer sequences, but at the cost of variance on shorter ones.

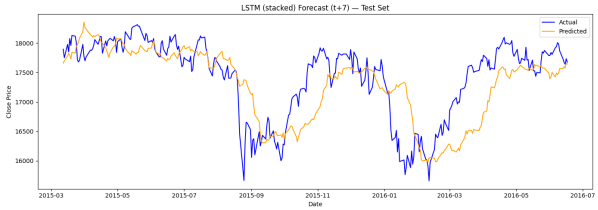

Fig. 6. Stacked LSTM Forecast — t+7 Horizon

The t+7 forecast illustrates that while the model roughly tracks price levels, deviations during large jumps show an inability to anticipate unexpected shocks—likely due to its deterministic nature despite behavioral inputs.

D. Temporal Convolutional Network (TCN)

TABLE IV
TCN TRAIN AND TEST PERFORMANCE

Horizon	Train			Test		
	R^2	RMSE	MAE	R^2	RMSE	MAE
t+1	0.9953	186.12	136.56	**0.8767**	**225.27**	**169.23**
t+3	0.9910	257.10	189.87	0.7584	316.20	231.83
t+7	0.9831	351.86	257.75	0.5483	434.50	331.63

The TCN model clearly outperformed all others at t+1 and t+3 horizons. Its dilated convolutions captured short and medium-term dependencies better than recurrent models. SHAP analysis further confirmed that intent and sentiment signals had stronger influence in these short-term scenarios.

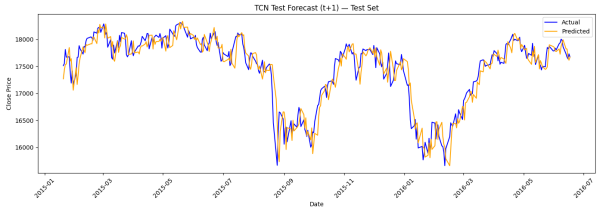

Fig. 7. TCN Forecast — t+1 Horizon

The t+1 TCN plot shows a tight fit between predictions and actual values, particularly around both local peaks and troughs. Better than ARIMA and LSTM, it adapted rapidly to directional shifts, validating the role of parallel temporal processing in financial forecasting.

VI. CONCLUSION

This study compared the effectiveness of ARIMA, LSTM (Simple and Stacked), and Temporal Convolutional Network (TCN) models for forecasting the DJIA closing price across three horizons: t+1, t+3, and t+7. Each model was trained using a multimodal dataset combining technical indicators with sentiment and intent signals derived from news and Reddit discussions.

Among all models, the TCN architecture demonstrated the strongest overall performance—especially at short-term horizons (t+1, t+3). Its ability to process sequential data using dilated causal convolutions allowed it to capture both rapid sentiment-driven fluctuations and broader temporal trends. ARIMA, while effective in t+1 forecasting due to its low-variance nature, failed to generalize beyond short horizons. LSTM models showed moderate improvement with added depth, but struggled with overfitting and volatility during long-term forecasts.

SHAP analysis further validated that behavioral features—particularly Reddit sentiment and urgency-related intents—were highly influential predictors, often outperforming traditional price-derived indicators in short-term contexts. This underlines the growing role of alternative data in financial time series modeling.

While TCN emerged as the most robust model in this study, its performance at t+7 still trailed behind ARIMA's simplicity in some cases. This reflects a broader trade-off between model complexity and horizon length: deep models learn more patterns, but may generalize poorly in low-signal regimes without careful tuning.

Future work may explore ensemble techniques that combine the stability of ARIMA, the sequence-learning strength of LSTM, and the parallelism of TCN. Integrating more granular features like intraday sentiment, volume spikes, or macroeconomic news could further improve forecasting accuracy. Additionally, interpretability tools like SHAP should remain central to understanding these black-box models in high-stakes financial environments.

ACKNOWLEDGMENT

The author utilized ChatGPT (OpenAI) in assisting with grammatical corrections and paraphrasing when compiling this research. Any final content was personally reviewed and edited for correctness by the author to ensure accuracy, coherence, and alignment with the research objectives [18].

REFERENCES

[1] C. E. Warburton and J. Pemberton, "Volatile financial conditions, assets prices and investment decisions: Analysis of daily data of djia and s&p500, from january to april of 2022," *Applied Econometrics and International Development*, vol. 23, no. 1, pp. 101–124, 2023.

[2] W. Ge, P. Lalbakhsh, L. Isai, A. Lenskiy, and H. Suominen, "Neural network–based financial volatility forecasting: A systematic review," *ACM Computing Survey*, vol. 55, no. 1, Jan. 2022, 213. [Online]. Available: https://doi.org/10.1145/3483596

[3] R. Bhowmik and S. Wang, "Stock market volatility and return analysis: A systematic literature review," *Entropy*, vol. 22, no. 5, 2020, h-Index: 101. [Online]. Available: https://www.mdpi.com/1099-4300/22/5/522

[4] P. Chen, Z. Boukouvalas, and R. Corizzo, "A deep fusion model for stock market prediction with news headlines and time series data," *Neural Computing and Applications*, vol. 36, pp. 21 229–21 271, 2024, h-Index: 130. [Online]. Available: https://doi.org/10.1007/s00521-024-10303-1

[5] A. Peivandizadeh, S. Hatami, A. Nakhjavani, L. Khoshsima, M. Reza Chalak Qazani, M. Haleem, and R. Alizadehsani, "Stock market prediction with transductive long short-term memory and social media sentiment analysis," *IEEE Access*, vol. 12, pp. 87 110–87 130, 2024.

[6] S. K. Bharti, P. Tratiya, and R. K. Gupta, "Stock market price prediction through news sentiment analysis & ensemble learning," in *2022 IEEE 2nd International Symposium on Sustainable Energy, Signal Processing and Cyber Security (iSSSC)*, 2022, pp. 1–5, h-Index: N/A. [Online]. Available: https://doi.org/10.1109/iSSSC56467.2022.10051623

[7] D. P. Gandhmal and K. Kumar, "Systematic analysis and review of stock market prediction techniques," *Computer Science Review*, vol. 34, p. 100190, 2019, h-Index: 75. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S157401371930084X

[8] K. Chen, Y. Zhou, and F. Dai, "A lstm-based method for stock returns prediction: A case study of china stock market," in *2015 IEEE International Conference on Big Data (Big Data)*, 2015, pp. 2823–2824, h-Index: 31. [Online]. Available: https://10.1109/BigData.2015.7364089

[9] J. Shah, R. Jain, V. Jolly, and A. Godbole, "Stock market prediction using bi-directional lstm," in *2021 International Conference on Communication information and Computing Technology (ICCICT)*, 2021, pp. 1–5, h-Index: N/A. [Online]. Available: https://10.1109/ICCICT50803.2021.9510147

[10] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv preprint arXiv:1803.01271*, 2018. [Online]. Available: https://arxiv.org/abs/1803.01271

[11] S. Gopali, F. Abri, S. Siami-Namini, and A. S. Namin, "A comparison of tcn and lstm models in detecting anomalies in time series data," in *2021 IEEE International Conference on Big Data (Big Data)*, 2021, pp. 2415–2420.

[12] J. Kumarappan, E. Rajasekar, S. Vairavasundaram *et al.*, "Federated learning enhanced mlp–lstm modeling in an integrated deep learning pipeline for stock market prediction," *International Journal of Computational Intelligence Systems*, vol. 17, p. 267, 2024, h-Index: 56. [Online]. Available: https://doi.org/10.1007/s44196-024-00680-9

[13] B. Jagadesh, N. RajaSekhar Reddy, P. Udayaraju *et al.*, "Enhanced stock market forecasting using dandelion optimization-driven 3d-cnn-gru classification," *Scientific Reports*, vol. 14, p. 20908, 2024, h-Index: 315. [Online]. Available: https://doi.org/10.1038/s41598-024-71873-7

[14] A. Churi, D. Chakraborty, R. Khatwani, G. Pinto, P. Shah, and R. Sekhar, "Stock price prediction using deep learning and sentiment analysis," in *2023 2nd International Conference on Futuristic Technologies (INCOFT)*, 2023, pp. 1–6, h-Index: N/A. [Online]. Available: https://doi.org/10.1109/INCOFT60753.2023.10425124

[15] M. Rhif, A. B. Abbes, B. Martínez, and I. R. Farah, "Veg-w2tcn: A parallel hybrid forecasting framework for non-stationary time series using wavelet and temporal convolution network model," *Applied Soft Computing*, vol. 137, p. 110172, 2023, h-Index: 190. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494623001904

[16] A. Raturi, K. G. Mula, K. Joshi, K. S. Sidhu, S. S. Chauhan, and V. Singh, "Comparative analysis of stock price prediction with multiple machine learning model," in *2025 3rd International Conference on Communication, Security, and Artificial Intelligence (ICCSAI)*, vol. 3, 2025, pp. 1428–1433.

[17] Kaggle, "Dow jones industrial average with news and reddit sentiment," https://www.kaggle.com/datasets/aaron7sun/stocknews/data, 2023, accessed: February 20, 2025.

[18] OpenAI, "Chatgpt," https://chat.openai.com, 2023, accessed: 20th February 2025. Used for summarization and content refinement.