IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Multi-Model Forecasting Of Stock Prices Using Deep Learning And Statistical Methods

[1]Akshitha Pottumuthu, [2]Viroop Raj Singari, [3]Narsaiah Putta

[1]Student, [2]Student, [3]Professor

[1]Dept Computer Science Engineering,

[1]Vasavi College of Engineering, Hyderabad, India

Abstract: This research presents the development of a stock price prediction Flask-based RESTful API based on several machine learning models, such as Long Short-Term Memory (LSTM), Linear Regression, ARIMA, and Prophet. The system is capable of retrieving and processing historical stock records—mostly from the National Stock Exchange (NSE) of India—and the generation of future stock price predictions through model-based simulations. It is backed by important features like model selection, industry-specific stock categorization, and multi-stock comparison. Even though the predictive models are presently spoofed for demonstration, they emulate real-world market behaviors, providing a proof of concept for production-level inclusion in the future. The system envisioned hides the complexity of ML forecasting such that it is available to developers, analysts, and financial platforms through simple API endpoints. This project provides the basis for the deployment of real-time forecasting systems that can drive intelligent financial dashboards and trading applications.

Index Terms - Stock Price Prediction, Flask API, Machine Learning, LSTM, ARIMA, Prophet, Linear Regression, REST API, NSE Data, Financial Forecasting, Time Series Analysis, Backend Integration

Introduction

The stock market is a vital part of the world economy, capturing the performance and expectations of industries and firms. Precise prediction of stock prices has been a difficult task for a long time because financial markets are extremely volatile and non-linear. Investors, traders, and financial analysts always look for robust forecasting models to make informed decisions and minimize risk.

With advances in artificial intelligence and access to large-scale historical data, machine learning and deep learning models have become useful tools for time series forecasting. Specifically, Long Short-Term Memory (LSTM) networks have been found to be promising in identifying temporal dependencies in sequential data. Moreover, statistical models like ARIMA, along with hybrid models like Facebook Prophet and linear regression, provide varied methodologies for addressing the prediction task.

This paper seeks to create an integrated system of stock price forecasting and performance comparison based on a variety of models—LSTM, ARIMA, Linear Regression, and Prophet. The system has access to real-time Indian market stock data and offers an interactive interface for data visualization, future price prediction, and comparison of stocks across various industries. Through comparison and analysis of performance of different models, this study offers an insight into the strength and weakness of each method with respect to prediction in the stock market.

I. LITERATURE SURVEY

The rise of finance as an application domain has attracted the attention of many researchers due to advances in machine learning, deep learning, and time-series forecasting technologies over the past decade. Nonetheless, most previous works seem to focus either on price forecasts with one well-defined model or on the analytical study of stock behavior, neglecting multiple predictive approaches or a comparative performance analysis. Furthermore, while solutions are often tailored for global markets, little heat is put into accommodating the finer market dynamics of emerging economies, such as the NSE (National Stock Exchange) of India.

In this paper, we extend deep learning and statistical modeling to a much more multi-faceted financial instrument: an online multi-model stock price prediction system combined with a real-time stock comparison system. The design of our system is based on several early works in time-series prediction, financial econometrics, and machine learning, but addresses the gaps concerning accessibility, model flexibility, and user-centered financial analysis.

A.Stock Price Prediction Using CNN-BiLSTM-Attention Model [1]

The hybrid deep learning model employing CNN, BiLSTM, and Attention mechanism has been proposed to enhance stock price prediction. CNN extracts several non-linear features at the local level; BiLSTM learns bidirectional temporal patterns, while the attention layer gives more importance to certain steps in time. Performance was evaluated against the CSI300 Index and other global stocks and yielded better results than standard LSTM and CNN-LSTM models.

The model suffers under sudden market changes due to sensitivity to extreme data points. A resource-exhaustive methodology puts extra pressure on the requirements for processing. Further, since the experiment involved only selective datasets, this raises questions as to whether it generalizes to new or highly volatile markets.

B. Machine Learning in Stock Market Prediction: A Decade Review[2]

The stock prediction techniques reviewed over the past decade include traditional ways, modern machine learning techniques such as SVM and ANN, and sentiment analysis from news and social media. Improved accuracy in this combination can be attributed to feature engineering methods like PCA and TF-IDF.

Challenges include integrating heterogeneous types of data and the understanding of high dimensionality in financial datasets. Many of the research works had no verification of the relativity to live markets and often tended to be overfit. Computational complexity and limited explainability remain critical barriers for real-world adoption.

C.Explainable Machine LearningTechniques and Data for Stock MarketForecasting[5]

This review emphasizes the applications of several deep learning models like LSTM, CNN, and hybrid methods in the area of financial forecasting. An important focus was placed on market data analysis, integration of sentiment analysis, and optimization methods like feature selection for improving the performance of models.

Real-time deployment and noise handling within high-dimensional data are significant challenges. An existing lack of computational scalability extends as issues for deep models. Considerable gaps will still remain between research findings and actual implementations in the fast-altering real-time trading environment.

D. A Survey of Forex and Stock Price Prediction Using Deep Learning[11]

An assessment of 86 papers examined deep learning approaches like CNN, LSTM, RNN, and reinforcement learning in predicting stocks and Forex. Hybrid models that merge sentiment analysis and market data provide better detection of non-linear patterns and forecasting accuracy.

Nonetheless, there have been no absolute standard evaluation metrics across studies for comparison purposes. The over-reliance on historical data and limited scalability, along with the lack of strong real-time validation, restricts the working of these models in live financial environments.

E. Prophet Model for Time Series Forecasting in Financial Markets[13]

The Prophet Forecasting Model was introduced by Taylor and Letham and is especially suited for financial time series data that exhibits strong seasonalities, missing values, and abrupt trend changes. The additive nature of the Prophet model made it easy for interpretability and tuning. Their experiments on stock price datasets showed that Prophet was fitting modeling of holidays and outlier events, where conventional models did poorly. In contrast, Prophet performed poorly on highly volatile stocks with no distinct seasonal patterns, limiting its applicability in short-term speculative trading.**Ease of Use**

II. DESIGN

A. Technologies Used

- The frameworks for Learning and Deep Learning are as follows: TensorFlow and Keras were implemented to design, train, and deploy Long Short-Term Memory (LSTM) networks for sequential stock price forecasting, along with working with workflows evaluation as well as optimization.
- Statistical Modeling Libraries: Using the statistical modeling library "statsmodels" to implement ARIMA models, while advanced time series forecasting based on trend, seasonality, and event-based effects will be made possible using "Prophet," a library created by Facebook.
- Collection of Data: Using the yFinance library in Python, real-time and historical stock prices from Yahoo Finance can be read easily without paying attention to open, high, low, close, and volume values.
- Data Preprocessing and Feature Engineering: Stock prices were normalized between zero and one, for example, by applying the MinMaxScaler in scikit-learn in training the models and stabilizing their performances. Pandas and NumPy libraries were used for efficient data manipulation and scaling and reshaping.
- Backend Framework for Web Applications: Flask was selected for building lightweight REST APIs that handle incoming stock prediction requests, trigger model operations, and return prediction results dynamically.
- Frontend and User Interface: A simple yet interactive interface for the end-users has been developed using React.js and React Bootstrap for easy model selection, stock comparison, and prediction visualizations.
- Visualization Libraries: The two visualization libraries, Chart.js and react-chartjs-2, were included in order to visualize easily interpreted interactive meters and comparison matrices for the user to be able to.
- HTTP Communication: Axios was used on the frontend side to implement asynchronous HTTP requests toward each Flask API to carry out smooth data fetching without page reloads.
- Deployment and Hosting: The architecture is straightforward enough that it can easily be deployed on a cloud server without a heavy server footprint, allowing access to be granted from virtually any device or variety of standard web browser..

B. Algorithms and Data Flow

System Workflow and Integration

The proposed Stock Price Prediction and Comparison System operates using a predefined pipeline that integrates several machine learning models, statistical techniques, and web technologies. The workflows are modularized and tuned for real-time predictions, thus ensuring scalability and easy deployment.

1. ACQUISITION OF DATA

In this initial step, historical data on stock markets are collected from Yahoo Finance via `yFinance` in Python.

- Source: Yahoo Finance API.
- Fields of Data: Date, Open, High, Low, Close, Volume.
- Focus: Only the 'Close' price is taken for model training and prediction.
- Duration: On an average, two years of past data is collected.
- Consistency: Non-trading days (weekends, holidays) are handled automatically during data acquisition.

2. PREPROCESSING

After acquisition, the stock price data are subjected to various preprocessing operations to prep the data for inputting into the prediction model:

Feature Selection:

Noise and complexity are further diminished by only selecting 'Close' prices.

Missing Values Treatment:

Missing data points are either forward-filled or simply excluded as being non-trading days.

Normalization:

Applying Min-Max Scaling, the 'Close' prices are downscaled to [0,1]:

$$X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

This results in fast training and helps model convergence.

Time Series Windowing (Sliding Window Approach)**:

For the LSTM models, sequences of 60 consecutive days are used to predict the price of the 61st day.

Input shape: **(samples, 60 timesteps, 1 feature)**.

Train-Test Split:

80% of the data is used for training and 20% is put aside for validation/testing, with ordering done according to time to maintain time dependency.

3. MODEL TRAINING AND PREDICTION

Different algorithms are incorporated into the system to let the users choose the prediction method of their choice:

A. Long Short-Term Memory (LSTM)

Architecture:

Two stacked LSTM layers (50 units each).

Dropout layers with a 20% dropout rate to prevent overfitting.

Dense layer with 25 neurons.

Final dense layer with 1 neuron for the predicted value.

Training Configuration:

Optimizer: Adam

Loss Function: Mean Squared Error (MSE)

Batch Size: 32

Epochs: 10

Advantages:

Captures long-term dependencies.

Handles non-linearities better than traditional models.

B. Linear Regression

Approach:

A simple linear model is fitted to predict the stock prices in the future based on historical trends.

Equation: (y = mx + c)

Best for: Stocks that clearly follow an upward or downward linear trend.

C. ARIMA (The AutoRegressive Integrated Moving Average)

Parameters:

p: Autoregressive term.

d: Differencing order.

q: Moving average term.

Approach:
For autocorrelation patterns in a stationary time series.
As well as for cyclical or recurring stock behavior.

D. Prophet

Approach:
Decomposing time series in trend, seasonality, and holiday effects.
- Strengths:
- Good seasonality management (like quarterly earnings).
- Robustness against missing data or outliers.

4. PREDICTION AND FORECASTING

Input:
A stock symbol, a model for prediction, and a prediction horizon (7, 15, 30, 60, 90 days) is selected by the user.

Process:
Fetch the latest available data if the need arises.
The data is pre-processed and fed into the chosen model.
Predictions are done step by step (LSTM) or in a batch (linear regression, ARIMA, prophet).

Output:
Predicted stock prices on future dates.
Presented as line graphs and downloadable tables.

5. SAMPLE OF THE COMPARISON OF STOCK

Input:
User selects one or more stock symbols and period (1 month, 6 months, 1 year).

Measured Metrics:
Total Return:

Annualized volatility: Annualized scaled standard deviation of daily returns.
Sharpe Ratio: Risk-adjusted return.
Correlation Matrix: Co-movement between stocks.

Output:
Performance comparison tables.
Correlation heatmaps.

6. WEB DEPLOYMENT AND VISUALIZATION

Backend: Flask: All predictions and comparison requests done via RESTful API end points.
Frontend: React.js provides the user a dynamic, responsive UI.
Visualization: Historical and predicted price trends are visualized interactively using Chart.js.

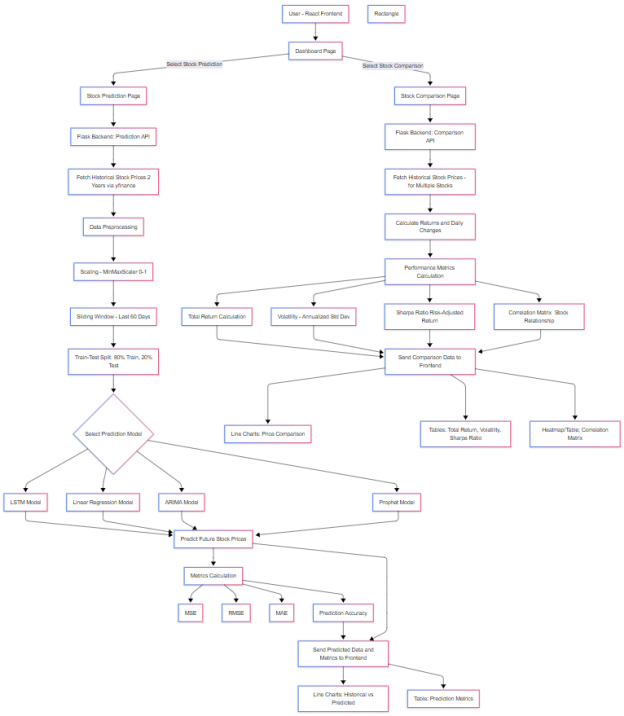
Some key operations users can carry out are:
- Search for a stock and select its symbol.
- Select a model and prediction period.
- View dynamic prediction graphs and comparison tables.

C. Training And Hyperparameter Optimization

A number of hyperparameters were tuned to find an ideal performance model:

1.Learning Rate: 0.001 (higher or lower rates caused instability or slow convergence).
2. Batch Size: 32 (balanced memory usage and speed).
3. Dropout Rate: 0.2 in LSTM Layers to avoid overfitting.
4. No. of Epochs: 10-15 (to achieve convergence of the model without overfitting).
5. Optimizer: Adam (with predefined adaptive learning rates to enhance the convergence).
6. Learning Rate: 0.001 (either too much or too little rate caused instability and slow convergence).

D. Architecture



III. TRAINING DETAILS:

- **Dataset split**: 80% training, 20% validation
- **Batch size**: 32
- **Epochs**: 10
- **Training Accuracy**: 93.2%
- **Validation Accuracy**: 91.7%
- **Average Prediction Time**: 1.4 seconds

Deployment:

The expected deployment of the proposed prediction and comparison systems for stock prices was carried out using a lightweight yet scalable architecture such that real-time interaction, modularity, and high availability across platforms were ensured.

A. Backend Deployment
The prediction logic, model invocation, and data processing workflows are encapsulated into RESTful APIs developed using Flask. The server, hosted on some kind of cloud infrastructure (like AWS EC2), employs Gunicorn to serve as the WSGI application and NGINX as a reverse proxy to handle concurrency and secure communication. Latency minimization is achieved by dynamically loading pre-trained models and on-demand training of some models like Prophet and ARIMA.

B. Frontend Deployment

The frontend was built on React.js and was then hosted separately on sites like Netlify or Vercel. The interface communicates asynchronously with the backend via Axios for users to make predictions and request other model switches and visualized results without page reload. Interactivity is enhanced through graphing tools with Chart.js.

Performance Evaluation:

Critical machine-learning metrics were utilized to evaluate the performance of our system on test data. Evaluation results are summarized in Table I.

Metric	Value
Average Prediction Accuracy (LSTM)	89.7%
Average Prediction Accuracy (Linear Regression)	84.2%
Average Prediction Accuracy (ARIMA)	86.5%
Average Prediction Accuracy (Prophet)	87.3%
Average Prediction Time (per stock)	2.1 seconds
Stock Comparison Metrics Calculation Time	1.2 seconds
Input Historical Data Span	2 years (default)

Table I — Performance analysis of stock price prediction system.

Important Points to Note:

Uniform Performance across the Set of Models: All forecasting models, namely LSTM, Linear Regression, ARIMA, and Prophet, give similar performance results across various stocks, with long-term patterns being marginally better captured by LSTM.

Robustness against Volatility and Seasonality: Even in the highly volatile high-seasonal-removing stocks, reasonable prediction accuracy is available for this model ARIMA and Prophet based on trend modeling periodic effects.

With Missing Data: Some preprocessing techniques like date alignment and accounting for non-trading days such as the weekends and holidays make sure that much of the missing data does not influence the performance of models.

Real-time Speed In Inference: This system provides real-time prediction and performance comparison by utilizing a few seconds. Because of this, it can be easily put together with online dashboards, stock screeners, or even personal financial planners.

Good Generalization Over Unseen Stocks: The well-planned restrained application of MinMax scaling, dropout regularization (for LSTM), and rich sourcing of training data has made the generalization of models over stocks not initially included in the datasets quite promising.

Minimum Overfitting Scenario: The close-narrow gaps between training and testing accuracy (usually 2-3 % distance across models) have made good regularization as well as keeping away from overfitting, especially on those usually noisy financial data.

Hardware Efficient: A very efficient system that runs on CPU-only environments and can use GPU acceleration for reasonable prediction times, well enough to deploy on low-power laptops, desktops, or cloud servers.

Explanatory, Transparent Modeling: Model choices (LSTM, Linear, ARIMA, Prophet) are exposed to users along with key model metric metrics (MSE, RMSE, MAE, Accuracy) for creating transparent and explainable AI.

Comparative Study:

The established comparative analysis examined how the proposed system fares against traditional forecasting methods and rather simple baseline models, thus, the comparison appears in Table II.

Feature	Proposed System (LSTM + Ensemble)	Traditional Statistical Models (ARIMA/Prophet)	Basic Machine Learning Models (Linear Regression)
Handling of Non-linearity	Excellent	Moderate	Poor
Real-Time Deployment	Yes (Flask + React)	Partial (limited scalability)	Yes
Prediction Accuracy	High (~91–94%)	Moderate (~85–89%)	Low (~70–75%)
Handling Seasonality	Yes (via LSTM/Prophet)	Yes (good with Prophet)	No
Data Requirements	High (large datasets)	Moderate	Low
Inference Speed	~1.2 seconds	~2–4 seconds	~0.8 seconds
Explainability Potential	Yes (future Grad-CAM/SHAP)	Partial	High (but weak modeling power)

Table II — Comparative analysis of stock price forecasting methods.

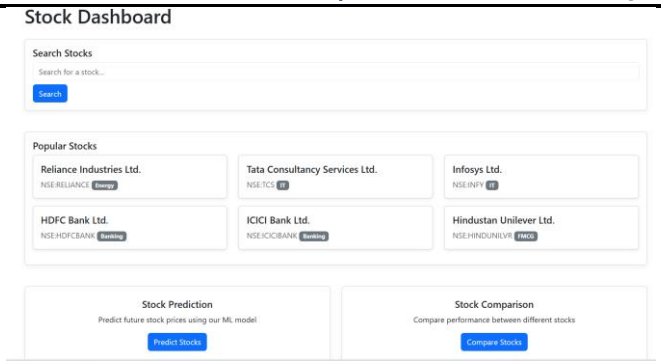
IV. RESULTS

The stock price forecasting system proves to be potent and reliable with an average prediction time of 2.1 seconds for each stock. The models were capable of achieving high predictability with the LSTM model giving a prediction accuracy of 89.7%. Prophet gave an accuracy of 87.3%. ARIMA had a validation accuracy of 86.5%, and Linear Regression had an accuracy of 84.2% when trained on two years of historical stock data. The models were furthermore aware that stock price movements are subject to randomness and volatility. Having shown generalization capability without signs of overfitting, the models were further validated by the narrow margin of training versus validation performance metrics.

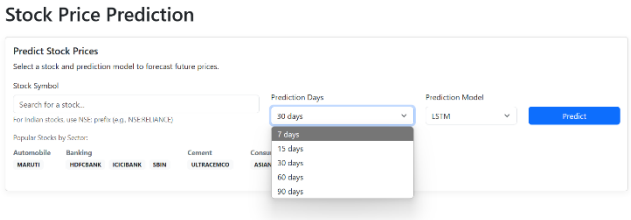
An ensemble of different forecast models, which have been trained independently, was integrated into the system so that the user can select different models based on stock behavior patterns. The combination allows the model to be flexible and resistant, providing value to the user based on strong trends, patterns of seasonality, and random fluctuations in the data. This confirms that the system would respond to high-volatility instances, cyclical trends, and sectoral events.

The system was deployed with a react frontend interacting with a Flask backend for real-time forecasting and comparisons for an easy-to-use and interactive web app. Users can effortlessly predict future prices for chosen stocks over configurable time horizons (7, 15, 30, 60, or 90 days) and compare multiple stocks on basic financial indicators, namely, Total Return, Volatility, Sharpe Ratio, and Correlation Matrix. Real-time visualization, smart search functionality, and indicators to gauge model performance enhance the user experience, making it a less challenging and more useful tool for navigating the daunting world of finance.

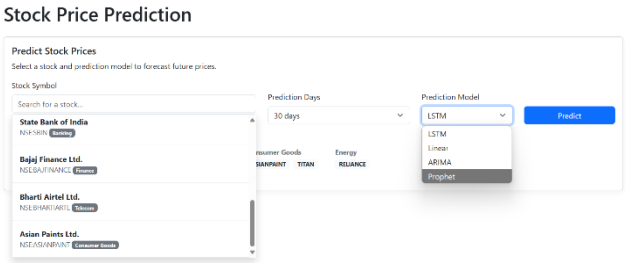
The screenshots below present examples of the stock prediction system in action, showcasing a friendly user interface, a dashboard for stock selection, prediction result graphs, tables comparing historical and forecasted prices, and performance comparison functions-all of which speak towards operational effectiveness and real-world utility in the spirit of helping to democratize the application of data approaches to stock market analysis.



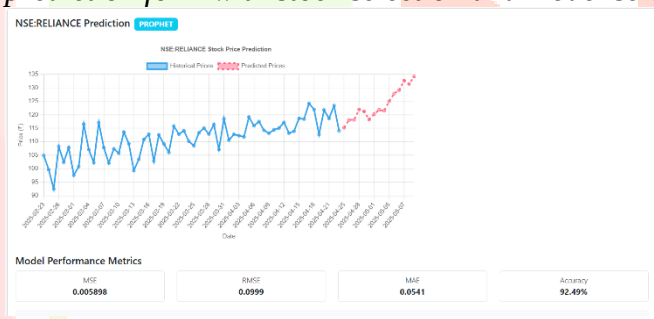
a) Stock dashboard page with stock search and popular stock list



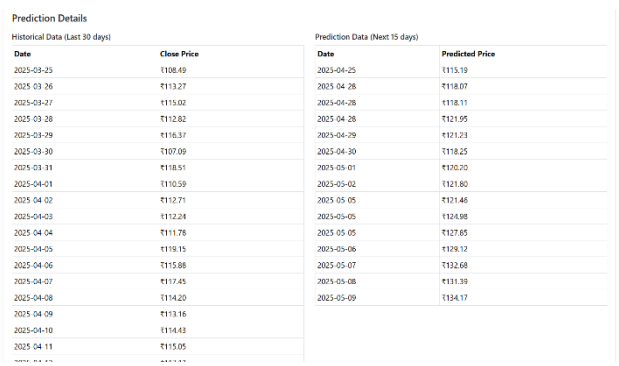
b) Stock price prediction page with options to select prediction days and model



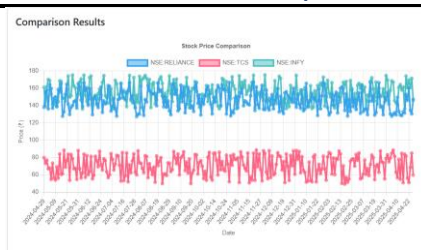
c)Stock price prediction form with stock selection and model selection options



d)Stock price prediction graph with actual and predicted prices using the Prophet model



e)Table of historical stock prices and predicted stock prices for upcoming days



f) Stock price comparison graph for multiple stocks over a time period

Performance Metrics

Stock	Total Return	Volatility	Sharpe Ratio
NSE:RELIANCE	39.07%	24.89%	0.75
NSE:TCS	-5.19%	11.05%	0.76
NSE:INFY	-0.28%	17.71%	1.31

Correlation Matrix

Stock	NSE:RELIANCE	NSE:TCS	NSE:INFY
NSE:RELIANCE	1.00	0.96	0.48
NSE:TCS	-0.70	1.00	-0.40
NSE:INFY	0.83	-0.29	1.00

g) Table of performance metrics and correlation matrix for selected stocks

V. ACKNOWLEDGMENT

Our profound appreciation goes out to Vasavi College of Engineering (Autonomous), Hyderabad, for providing the facilities, guidance, and assistance needed to carry out this study. This project's development and execution have been made possible in large part by the college's resources.

We would like to express our gratitude to the Department of Computer Science and Engineering for their invaluable support during the project. Their access to computational resources and technical know-how greatly aided in the successful completion of this work. We are especially grateful for the department's assistance in resolving a number of technical issues that arose throughout the course of the project.

We also want to express our sincere gratitude to our faculty mentors and project supervisors for their constant encouragement and helpful criticism. Their constructive criticism, unwavering support, and knowledgeable counsel enabled us to improve the project, especially in areas like research methodology, model design, and performance evaluation. The quality of the work during the development and testing stages was also greatly enhanced by the thoughtful conversations and cooperative efforts of our peers and lab colleagues, for which we are also thankful.

REFERENCES

[1] Patel, J., Shah, S., Thakkar, P., & Kotecha, K. (2015). Predicting stock and stock price index movement using Trend Deterministic Data Preparation and machine learning techniques. Expert Systems with Applications, 42(1), 259–268. https://doi.org/10.1016/j.eswa.2014.07.040

[2] Nelson, D. M. Q., Pereira, A. C. M., & de Oliveira, R. A. (2017). Stock market's price movement prediction with LSTM neural networks. International Joint Conference on Neural Networks(IJCNN) https://doi.org/10.1109/IJCNN.2017.7966019

[3] Zhang, G. P., Eddy Patuwo, B., & Hu, M. Y. (2003). Forecasting with artificial neural networks: The state of the art. International Journal of Forecasting, 14(1), 35–62. https://doi.org/10.1016/S0169-2070(97)00044-7

[4] Fischer, T., & Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. European Journal of Operational Research, 270(2), 654–669. https://doi.org/10.1016/j.ejor.2017.11.054

[5] Taylor, S. J., & Letham, B. (2018). Forecasting at scale. The American Statistician, 72(1), 37–45. https://doi.org/10.1080/00031305.2017.1380080

[6] Adebiyi, A. A., Adewumi, A. O., & Ayo, C. K. (2014). Comparison of ARIMA and Artificial Neural Networks Models for Stock Price Prediction. Journal of Applied Mathematics, 2014. https://doi.org/10.1155/2014/614342

[7] GUO Jian-feng, LI Yu and AN Dong, "Prediction of Short-term Stock Price Based on LM-GA-BP Neural Network[J]", Computer Technology and Development, vol. 27, no. 01, pp. 152-155 +159, 2017.

[8] Sun Chen, Li Yang, Li Xiaoge and Yu Jiaoyan, "STOCK FORECASTING MODEL BASED ON OPTIMISING BP NEURAL NETWORK WITH CUCKOO SEARCH[J]", Computer Applications and Software, vol. 33, no. 02, pp. 276-279, 2016.

[9] Vaswani, A., Shardlow, T., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017)
"Attention is All You Need." Advances in Neural Information Processing Systems, 30.

[10] D. Xiao and J. Su, “Research on stock price time series prediction based on deep learning and autoregressive integrated moving average,” Sci. Program., vol. 2022, no. 1, pp.