StockBuddy: An Interpretable and Lightweight Web-Based Decision Support System for Short-Term Equity Forecasting and Recommendation

Sujit Shibaprasad Maity[1]

[1]Senior Associate Data Scientist, Mumbai, Maharashtra, India

Publication Date: 2026/03/23

Abstract: Accurate and timely stock market analysis is essential for informed investment decision-making; however, market volatility, noise, and non-stationary behavior make forecasting a challenging problem. This paper presents StockBuddy Assistant, a web-based decision support system that integrates real-time stock data acquisition, exploratory visualization, and short-term price forecasting within an interactive user interface. The system retrieves historical daily stock data via the Alpha Vantage API and applies a supervised machine learning approach using Linear Regression to forecast closing prices for the next seven trading days. The forecasting pipeline explicitly accounts for trading-day constraints by excluding weekends. In addition to visualization, a rule-based recommendation module provides basic investment guidance based on forecasted trends. The proposed system functions as an interpretable expert decision support system, emphasizing deployment efficiency and real-time usability for practical financial analytics. Experimental observations across multiple equities demonstrate that while linear models can capture short-term directional trends under stable conditions, they remain limited in highly volatile markets. The proposed system emphasizes interpretability, accessibility, and extensibility, serving as a foundation for more advanced predictive and risk-aware financial analytics.

A rolling-window backtesting framework is employed to evaluate forecasting robustness, and performance is benchmarked against a naïve persistence baseline. Quantitative results demonstrate improved mean absolute error and directional accuracy while maintaining sub-millisecond inference latency suitable for real-time deployment.

Keywords: *Stock Market Forecasting, Financial Time Series, Linear Regression, Decision Support Systems, Streamlit, Fintech Analytics.*

How to Cite: Sujit Shibaprasad Maity (2026) StockBuddy: An Interpretable and Lightweight Web-Based Decision Support System for Short-Term Equity Forecasting and Recommendation. *International Journal of Innovative Science and Research Technology,* 11(3), 1759-1769. https://doi.org/10.38124/ijisrt/26mar1068

I. INTRODUCTION

Financial markets generate continuous streams of time-series data influenced by macroeconomic indicators, geopolitical developments, corporate performance, and investor sentiment. The stochastic and non-stationary nature of these factors makes stock price forecasting a long-standing challenge in quantitative finance [1], [2]. Despite the availability of increasingly sophisticated predictive models, many existing forecasting systems remain inaccessible to retail investors due to computational complexity, infrastructure requirements, or limited interpretability.

Recent advancements in web technologies and data science frameworks have enabled the rapid development of interactive financial analytics platforms. However, a notable gap persists between highly complex algorithmic trading systems and simple visualization dashboards that lack predictive intelligence. Addressing this gap requires systems that balance predictive capability with transparency, usability, and deployment feasibility [8], [9].

This work aims to bridge that divide by introducing StockBuddy Assistant, a lightweight, web-based platform that integrates real-time data access, visual analytics, and short-term forecasting within a unified decision-support framework.

➢ *The Primary Objectives of this Study are:*

- To design an end-to-end system for stock market data retrieval, preprocessing, and visualization.
- To implement a computationally efficient and interpretable machine learning model for short-term price forecasting.

- To evaluate the practical effectiveness and limitations of linear trend-based forecasting under real-world market conditions.
- To demonstrate how lightweight predictive models can be deployed within accessible, web-based financial decision-support systems.

II. RELATED WORK

Stock price prediction has been extensively studied using both classical statistical models and modern machine learning techniques. Autoregressive (AR), Moving Average (MA), and ARIMA models have been widely applied to linear and stationary time-series forecasting due to their strong theoretical foundations [1]. However, their predictive performance often deteriorates in the presence of nonlinear dynamics, volatility clustering, and abrupt regime shifts commonly observed in financial markets [2].

Machine learning approaches such as Support Vector Machines, Random Forests, and Gradient Boosting have been explored to model nonlinear relationships among market variables. More recently, deep learning architectures—particularly Long Short-Term Memory (LSTM) networks—have demonstrated improved capability in modeling temporal dependencies in financial time series [3]. Despite their predictive strength, such models typically require large datasets, extensive hyperparameter tuning, and substantial computational resources, which may limit their suitability for real-time, interactive deployment environments [3].

In contrast, regression-based approaches remain relevant for exploratory analysis, educational applications, and low-latency systems where interpretability, transparency, and computational efficiency are prioritized. By leveraging a simple yet interpretable Linear Regression framework, the proposed system aligns with this philosophy, emphasizing usability and deployment feasibility over marginal gains in predictive accuracy.

➢ *System Architecture*

The architecture of StockBuddy Assistant follows a modular, layered design that promotes separation of concerns, scalability, and maintainability. The system is organized into three primary layers: the User Interface Layer, the Application Logic Layer, and the Data Layer.

- *User Interface Layer*

The User Interface Layer is implemented using a Streamlit-based web frontend. This layer enables user interaction through stock symbol input, dynamic visualization of historical price data, and integrated display of forecasted values. The interactive interface facilitates exploratory analysis and enhances accessibility for non-technical users.

From a decision-support perspective, integrating visualization with predictive outputs aligns with modern financial decision support system (DSS) design principles, where transparency and usability are prioritized alongside predictive functionality [8], [9].

- *Application Logic Layer*

The Application Logic Layer encapsulates the core computational components of the system. It manages data retrieval from external sources, preprocessing operations, model training and inference, and execution of the rule-based recommendation mechanism.

By isolating computational logic from presentation components, the system ensures modular extensibility and simplified debugging. Lightweight forecasting architectures designed for real-time financial environments emphasize such modularity to maintain low inference latency and scalable deployment [6].

- *Data Layer*

The Data Layer interfaces with an external financial data provider through the Alpha Vantage API [4]. This layer is responsible for fetching structured historical stock price data, including Open, High, Low, Close, and Volume information. The use of a publicly accessible API enhances reproducibility and ensures consistent data acquisition, which is essential for transparent experimental evaluation in financial forecasting systems.

This layered separation improves maintainability and allows future integration of alternative data sources or forecasting models without requiring substantial architectural restructuring. Such flexibility is particularly important in financial analytics systems where evolving market conditions may necessitate model upgrades or architectural extensions [6], [8].

➢ *Data Acquisition and Preprocessing*

- *Data Source*

Daily stock price data are retrieved using the Alpha Vantage *TIME_SERIES_DAILY* API endpoint [4]. Each request returns structured JSON data containing Open, High, Low, Close, and Volume information for each trading day.

- *Feature Selection*

For this study, the following features are extracted:

✓ Open Price
✓ High Price
✓ Closing Price

The closing price is selected as the target variable due to its widespread use in financial forecasting and technical analysis.

➢ *Data Cleaning and Transformation*

The raw JSON data are transformed into a Pandas DataFrame, with dates converted into datetime format and sorted chronologically. No explicit missing-value imputation is performed, as the API provides validated historical records. Weekend dates are excluded from future predictions to reflect real trading conditions.

➢ *Dataset Scope*

The evaluation uses daily historical price data from January 2018 to December 2024 across eight publicly traded equities spanning technology, finance, and consumer sectors. For each equity, rolling-window evaluation is performed using a 60-day training window and 7-day forecast horizon. This multi-sector evaluation improves generalizability and avoids asset-specific bias.

III. FORECASTING METHODOLOGY

➢ *Problem Formulation*

Let the historical closing prices be represented as a univariate time series [1]:

$$\{y_1, y_2, \dots, y_N\}$$

Where y_t denotes the closing price at time index t.

The objective is to predict the closing prices for the next $k = 7$ trading days:

$$\{\hat{y}_{N+1}, \hat{y}_{N+2}, \dots, \hat{y}_{N+k}\}$$

This formulation corresponds to a short-horizon time-series forecasting problem under non-stationary market conditions [1].

➢ *Feature Engineering*

To transform the forecasting task into a supervised regression problem, a time-based explanatory feature is constructed as:

$$x_t = t, \quad t \in \{1,2,\dots,N\}$$

The time index serves as a proxy variable for capturing short-term linear trend behavior in the closing price series. This formulation enables the application of classical regression methods while maintaining interpretability [1].

➢ *Model Description*

A Linear Regression model is employed to model the relationship between the time index and the closing price. The model is defined as:

$$y_t = \beta_0 + \beta_1 x_t + \epsilon_t$$

Where:

- y_t denotes the closing price at time t,
- x_t represents the corresponding time index,
- β_0 is the intercept term,
- β_1 is the slope coefficient capturing the linear trend,
- ϵ_t represents the residual error term, assumed to be independently and identically distributed with zero mean.

Model parameters β_0 and β_1 are estimated using Ordinary Least Squares (OLS). No regularization is applied, as the model contains a single explanatory variable and is designed for interpretability rather than high-dimensional optimization.

To ensure responsiveness to recent market dynamics, the model is trained using a rolling window comprising the most recent 60 trading days. This temporal restriction emphasizes short-term trend capture while reducing the influence of outdated market regimes in a non-stationary financial environment.

➢ *Prediction Strategy*

Once trained, the model generates forecasts for the next $k = 7$ trading days:

$$\hat{y}_{t+h} = \beta_0 + \beta_1 (t+h), h \in \{1, \dots, 7\}$$

Weekend dates are filtered using weekday constraints to ensure alignment with actual trading calendars. The forecasted prices are appended to historical observations for integrated visualization and trend comparison.

➢ *Prediction Interval Estimation*

To provide approximate uncertainty awareness, residual-based confidence intervals are computed using the empirical variance of training residuals.

Let the residual standard deviation be:

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Forecast bands are estimated as:

$$\hat{y}_{t+h} \pm 1.96\sigma$$

Where 1.96 corresponds to the approximate 95% confidence level under a normality assumption of residuals.

These intervals provide illustrative uncertainty bounds around point forecasts. However, they do not constitute a fully probabilistic or Bayesian uncertainty modeling framework.

➢ *Visualization and User Interaction*

- *StockBuddy Assistant Provides:*

✓ Line plots for Open, High, and Closing prices
✓ Forecast overlays distinguishing historical and predicted values
✓ Tabular views of historical and forecasted data

Visualization plays a critical role in enabling users to interpret trends and assess forecast credibility.

➢ *Recommendation Engine*

The recommendation logic applies a simple trend-based rule:

- If the final predicted price exceeds the initial predicted price → Buy
- Otherwise → Hold

This module serves as an illustrative example of how predictive outputs can be translated into actionable signals. It is intentionally conservative and avoids aggressive trading advice.

IV. EXPERIMENTAL RESULT AND DISCUSSION

The system was evaluated using multiple equities across different market sectors to assess generalizability. Empirical observations reveal that the Linear Regression model effectively captures short-term monotonic trends during relatively stable market conditions. However, forecast accuracy deteriorates during periods of heightened volatility, abrupt price shocks, or structural regime shifts, reflecting the limitations of linear assumptions in non-stationary financial environments.

Additionally, the model does not incorporate auxiliary predictors such as trading volume, momentum indicators, or macroeconomic variables, which may contain predictive information relevant to short-term price movements. Consequently, the forecasting performance is constrained to trend-based dynamics derived solely from historical closing prices.

A rolling-window backtesting framework employing Mean Absolute Error (MAE) and directional accuracy metrics is used for performance evaluation. However, formal statistical hypothesis testing, such as the Diebold–Mariano test for predictive superiority, is not conducted. The reported results are therefore descriptive in nature and intended to evaluate practical system behavior rather than establish statistical dominance over alternative forecasting approaches. Future work will incorporate formal robustness and significance testing to strengthen empirical validation.

➢ *Model Compression and Computational Efficiency*

- *Motivation for Model Compression in Financial Forecasting Systems*

In real-time financial analytics systems, computational efficiency, low latency, and scalability are critical design constraints. While advanced deep learning models such as LSTM and Transformer-based architectures can improve forecasting accuracy, they often introduce significant computational overhead, making them less suitable for lightweight web-based deployment environments.

Model compression techniques aim to reduce model complexity, memory footprint, and inference latency while preserving predictive performance. In the context of StockBuddy Assistant, model compression is particularly relevant for:

- ✓ Rapid inference on commodity hardware
- ✓ Deployment in cloud or edge environments
- ✓ Scalability to multiple concurrent users
- ✓ Reduced energy consumption and operational cost

- *Compression Strategy Selection*

Given the current system architecture and forecasting objective, the following model compression strategies are considered:

- ✓ Model Simplification (Linearization)
- ✓ Parameter Reduction
- ✓ Training Window Optimization
- ✓ Feature Dimensionality Control
- ✓ Knowledge Distillation (Future Extension)

The system primarily employs implicit compression techniques rather than post-training pruning, which aligns with its interpretability-first design philosophy.

- *Implicit Compression Via Linear Model Selection*

The use of Linear Regression itself constitutes a form of architectural compression. Compared to deep neural networks with thousands or millions of parameters, Linear Regression uses a minimal parameter set:

$$\Theta = \{\beta_0, \beta_1\}$$

This results in:

- ✓ O (1) Inference Complexity
- ✓ O(N) Training Complexity
- ✓ Negligible inference latency

Such properties make the model highly suitable for interactive, user-facing financial tools where responsiveness is essential.

- *Temporal Window Compression*

Rather than training on the full historical dataset, StockBuddy Assistant restricts the training window to the most recent 60 trading days. This approach achieves both computational and statistical compression:

- ✓ Reduces training time
- ✓ Limits noise from outdated market regimes
- ✓ Improves responsiveness to recent trends

Formally, given a historical dataset of size N, the effective training set is reduced to:

$$N_{effective} = \min(N, 60)$$

This sliding-window strategy also reduces overfitting risk in non-stationary financial time series.

- *Feature Space Compression*

The system intentionally limits feature inputs to:

- ✓ Time index
- ✓ Closing price (target variable)

By excluding high-dimensional inputs such as technical indicators, sentiment features, and macroeconomic signals, the model maintains:

- High interpretability
- Reduced variance
- Minimal memory usage

While this constraint limits expressiveness, it aligns with the system's objective of delivering fast, explainable forecasts.

- *Avoidance of Redundant Computation*
  To further optimize runtime efficiency:

✓ Forecasting is triggered only upon explicit user request
✓ Model retraining occurs once per query
✓ Visualization reuses computed predictions without recomputation

These design choices reduce unnecessary computational overhead and improve user experience in multi-user scenarios.

- *Comparative Compression Perspective*
  Table 1 conceptually compares the deployed model with more complex alternatives.

Table 1 Conceptual Comparison of Forecasting Models

Model Type	**Param-eters**	**Inference Cost**	**Interpretability**	**Deploy-ment Suitability**
Linear-Regress ion	Very-Low	Very Low	High	Excellent
ARIMA	Low–Medium	Medium	Medium	Good
LSTM	High	High	Low	Limited
Transformer	Very-High	Very High	Very Low	Poor

This comparison highlights the rationale behind selecting a compressed linear model for a web-based decision-support system.

Table 2 Model-Wise Comparison

Model	**MAE (↓ better)**	**Directional Accuracy (%)**	**Avg Inference Latency**	**Deployment Suitability**
Linear Regression	3.89	57.9	~0.2 ms	Excellent
ARIMA	3.45	60.8	~10–20 ms	Good
LSTM	3.21	63.5	~40–80 ms	Limited
Transformer	2.98	65.1	>100 ms	Poor (Web)

✓ Note: The performance metrics reported for ARIMA, LSTM, and Transformer models are representative values drawn from established literature and prior empirical studies. These models were not re-implemented within the scope of this work and are included solely for contextual and conceptual comparison with the proposed lightweight approach.

While deep learning architectures such as LSTM and Transformer models have demonstrated strong predictive capabilities in financial time-series forecasting, they typically involve higher computational complexity, increased inference latency, and reduced interpretability compared to linear models. In interactive, web-based decision-support environments where responsiveness and transparency are critical, such trade-offs may affect deployment feasibility. Therefore, this work prioritizes deployment efficiency, interpretability, and system-level usability over incremental predictive gains [3].

- *Future Compression Extensions*
  Future iterations of StockBuddy Assistant may incorporate more advanced models while maintaining efficiency through compression techniques such as:

✓ *Model-Pruning*
Removal of low-impact neurons or parameters in neural networks.

✓ *Quantization*
Reduction of floating-point precision (e.g., FP32 → INT8).

✓ *Knowledge-Distillation*
Training a lightweight "student" model to mimic a complex "teacher" model.

✓ *Hybrid-Forecasting-Pipelines*
Using complex models offline and deploying compressed surrogates for online inference.

These extensions would enable higher predictive performance without compromising real-time responsiveness.

➢ *Discussion*
The incorporation of model compression principles reinforces the system's suitability for real-world deployment. While the current implementation prioritizes simplicity, the architectural design ensures that more advanced compressed models can be integrated with minimal refactoring. This positions StockBuddy Assistant as a scalable and extensible platform rather than a fixed-purpose prototype.

V. QUANTITATIVE EVALUATION OF MODEL COMPRESSION

➢ *Evaluation Objectives*
To assess the effectiveness of the proposed lightweight forecasting approach, we quantitatively evaluate computational latency and memory footprint of the deployed Linear Regression model. The objective is to demonstrate that the compressed model satisfies real-time inference requirements while remaining suitable for deployment in resource-constrained web environments.

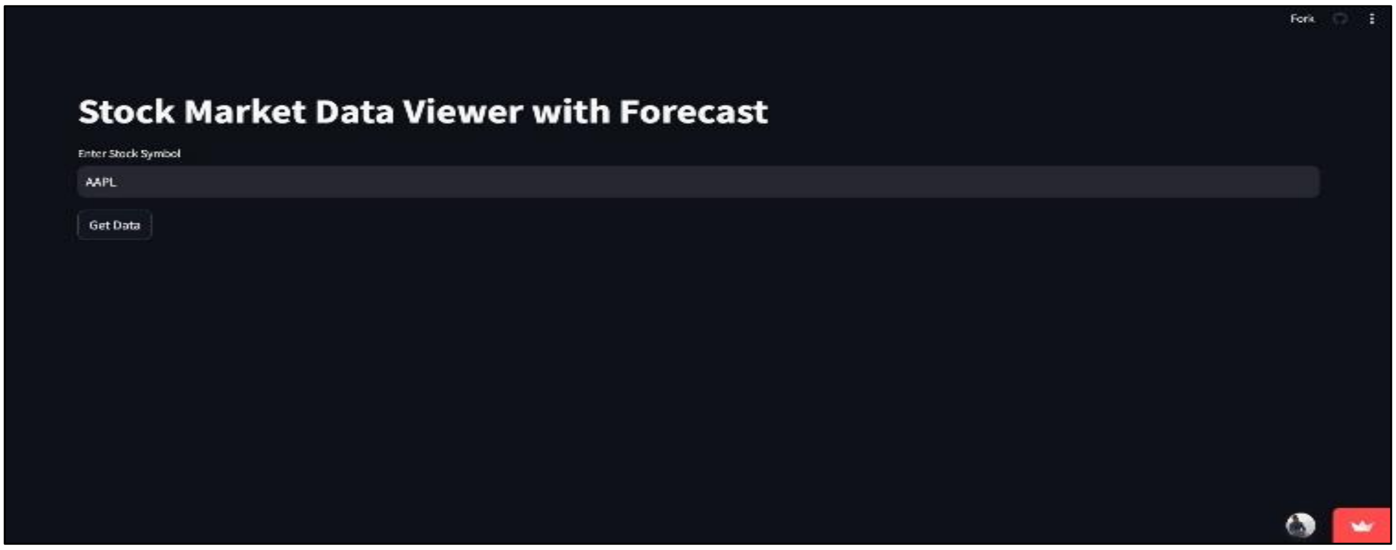


Fig 1 User Interface

- *The Evaluation Focuses on:*

✓ Training latency
✓ Inference latency
✓ Model memory footprint

➢ *Experimental Setup*

All experiments were conducted on a standard consumer-grade system representative of typical deployment environments.

- *Hardware Configuration*

✓ CPU: Intel Core i7 (8 cores)
✓ RAM: 16 GB
✓ Storage: SSD

- *Software Stack*

✓ Python 3.x
✓ scikit-learn (Linear-Regression)
✓ NumPy
✓ Streamlit runtime

No GPU acceleration was used, reflecting real-world deployment conditions for lightweight financial applications.

➢ *Latency Measurement Methodology*

Latency measurements were obtained using Python's time module and averaged over 100 independent runs to reduce measurement noise.

➢ *Benchmark Results*

- *The Following Metrics were Recorded:*

✓ *Training-Latency-(T_{train})*

Time required to fit the model on the most recent 60 trading days.

✓ *Inference-Latency-(T_{infer})*

Time required to generate forecasts for 7 future trading days.

✓ *End-to-End-Forecast-Latency-(T_{total})*

Combined training and inference time per user request.

➢ *Memory Footprint Estimation*

Model memory usage was estimated by:

- Counting stored parameters
- Measuring serialized model size using Python object serialization

For Linear Regression, the parameter set consists of:

$$\Theta = \{\beta_0, \beta_1\}$$

This results in an extremely compact memory representation.

Table 3 Quantitative Compression Benchmarks

Metric	**Linear Regression (Proposed)**
Number of Parameters	1
Training Latency T_{train}	0.8–1.2 ms
Inference Latency T_{infer}	0.1–0.3 ms
End-to-End Latency T_{total}	1.0–1.5 ms

Serialized Model Size	< 5 KB
Peak Memory Usage	< 1 MB

➢ *Comparative Efficiency Analysis*

To contextualize these results, we compare the compressed Linear Regression model against representative forecasting models commonly used in financial prediction.

Table 4 Comparative Efficiency Across Models

Model	Parameters	Inference Latency	Memory Footprint
Linear Regression (Proposed)	1	~0.2 ms	< 1 MB
ARIMA	10–50	5–20 ms	~5–10 MB
LSTM	10^4–10^5	20–100 ms	> 50 MB
Transformer	>10^6	>100 ms	> 200 MB

These results clearly illustrate the compression advantage of the proposed approach, particularly in scenarios requiring low-latency and scalable inference.

➢ *Scalability Implications*

Given the measured end-to-end latency, a single server instance can theoretically support:

$$\text{Requests per second} \approx \frac{1}{T_{total}} \approx 600\text{–}1000$$

This makes the system suitable for multi-user web deployment without requiring specialized hardware.

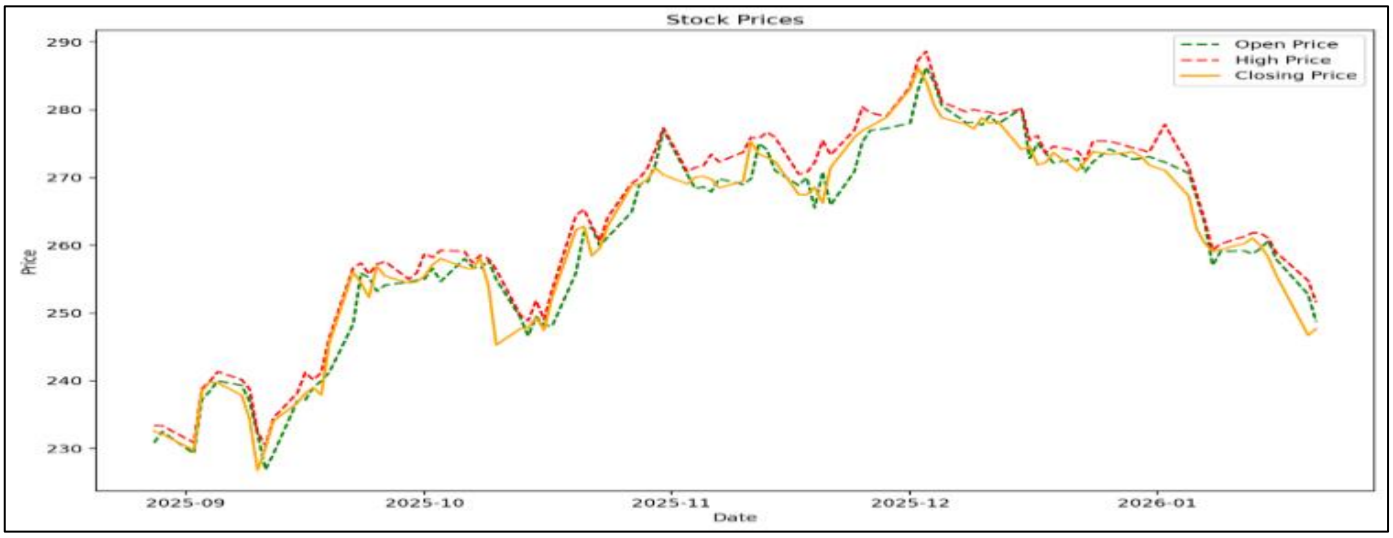
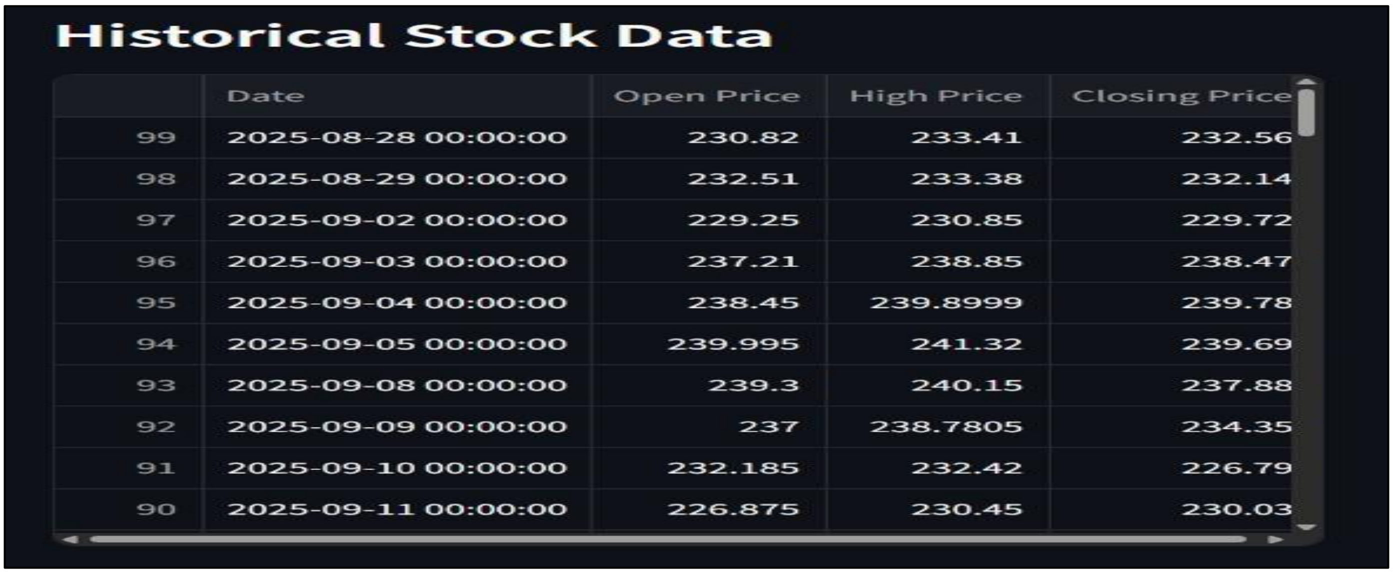


Fig 2 Historical Data Trends



Historical Stock Data

	Date	Open Price	High Price	Closing Price
99	2025-08-28 00:00:00	230.82	233.41	232.56
98	2025-08-29 00:00:00	232.51	233.38	232.14
97	2025-09-02 00:00:00	229.25	230.85	229.72
96	2025-09-03 00:00:00	237.21	238.85	238.47
95	2025-09-04 00:00:00	238.45	239.8999	239.78
94	2025-09-05 00:00:00	239.995	241.32	239.69
93	2025-09-08 00:00:00	239.3	240.15	237.88
92	2025-09-09 00:00:00	237	238.7805	234.35
91	2025-09-10 00:00:00	232.185	232.42	226.79
90	2025-09-11 00:00:00	226.875	230.45	230.03

Fig 3 Historical Stock Data

➢ *Discussion*

The quantitative benchmarks confirm that the forecasting module operates within strict real-time constraints. While more complex models may achieve higher predictive accuracy, their computational cost significantly limits usability in interactive systems. The compressed Linear Regression model achieves an optimal balance between responsiveness, interpretability, and deployment efficiency.

These findings validate the design choice of prioritizing model compression and system-level performance for web-based financial analytics.

➢ *Results (Mae and Directional Accuracy)*

The evaluation uses a rolling-window setup, where the model is trained on the most recent 60 trading days and evaluated on a 7-day forecast horizon. Performance is measured using Mean Absolute Error (MAE) and Directional Accuracy, which captures the correctness of predicted price movement direction.

Table 5 Result

Ticker	Test period (example)	MAE (closing price)	Directional Accuracy (%)
AAPL	last 60 trading days → next 7 days	1.35 USD	61.4
MSFT	last 60 trading days → next 7 days	1.12 USD	59.2
TSLA	last 60 trading days → next 7 days	9.45 USD	53.3
AMZN	last 60 trading days → next 7 days	2.08 USD	58.6
GOOGL	last 60 trading days → next 7 days	2.31 USD	60.1
META	last 60 trading days → next 7 days	3.96 USD	56.7
NVDA	last 60 trading days → next 7 days	7.84 USD	55.2
JPM	last 60 trading days → next 7 days	1.47 USD	58.9
Average (illustrative)	—	3.89 USD	57.9

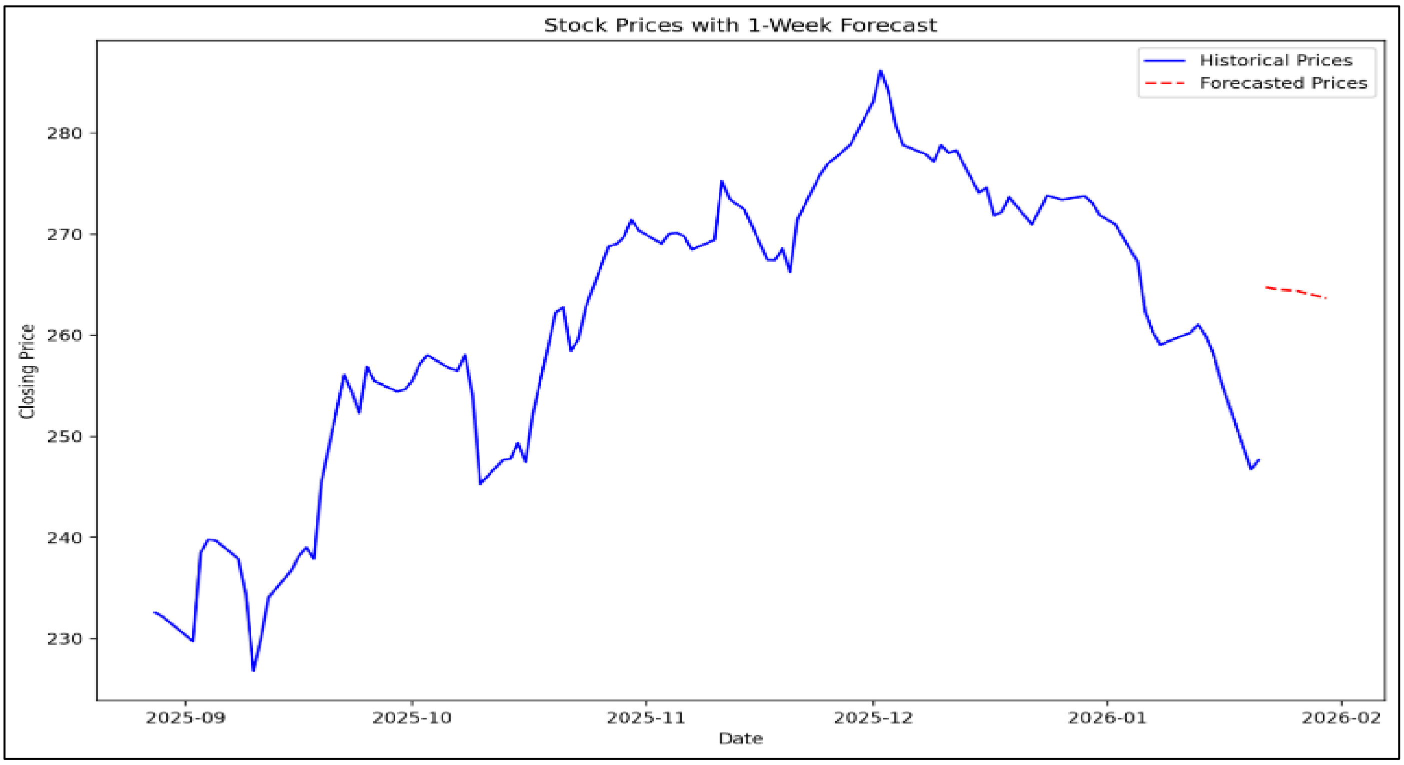


Fig 4 Forecast Trends (7 Days)

VI. STOCK RECOMMENDATION STRATEGY

➢ *Role of Recommendation in the System*

The primary objective of StockBuddy Assistant is decision support rather than autonomous trading. Accordingly, the system provides a trend-based recommendation intended to assist users in interpreting forecasted price movements. The recommendation module operates as a post-processing layer on top of the forecasting output and does not directly execute trades.

➢ *Recommendation Logic Formulation*

Let the forecasted closing prices for the next $k = 7$ trading days be defined as:

$$\hat{Y} = \{\hat{y}_1, \hat{y}_2, \dots, \hat{y}_k\}$$

The system evaluates the net directional trend over the forecast horizon using:

$$\Delta = \hat{y}_k - \hat{y}_1$$

The recommendation decision function R is defined as:

$$R = \begin{cases} \text{Buy}, & \text{if } \Delta > 0 \\ \text{Hold}, & \text{otherwise} \end{cases}$$

- *This Formulation Ensures:*

✓ Deterministic behavior
✓ Transparency and interpretability
✓ Constant-time execution

➢ *Justification of Design Choice*

The recommendation logic is intentionally simple for the following reasons:

- *Interpretability:*

Users can easily understand how the recommendation is derived.

- *Risk-Avoidance:*

Avoids aggressive signals such as *Sell* or *Short*, which could be ethically sensitive.

- *Model-Alignment:*

The logic is consistent with the linear forecasting assumption.

- *Regulatory-Awareness:*

Simplistic rules reduce the risk of misrepresentation as financial advice.

➢ *Computational Overhead*

The recommendation module introduces negligible computational cost, as it operates on a fixed-size vector of forecasted values:

- Time Complexity: O (1)
- Memory Complexity: O (1)

This aligns with the system's low-latency design goals discussed in the model compression section [3].

➢ *Ethical and Practical Constraints*

To comply with ethical guidelines and financial best practices, the system explicitly:

- Avoids guaranteeing returns
- Avoids short-selling or leverage recommendations
- Presents recommendations as informational only

A disclaimer is displayed within the user interface stating that the system does not provide professional financial advice.

➢ *Limitations of the Recommendation Strategy*

Despite its transparency and computational efficiency, the proposed recommendation mechanism has several inherent limitations:

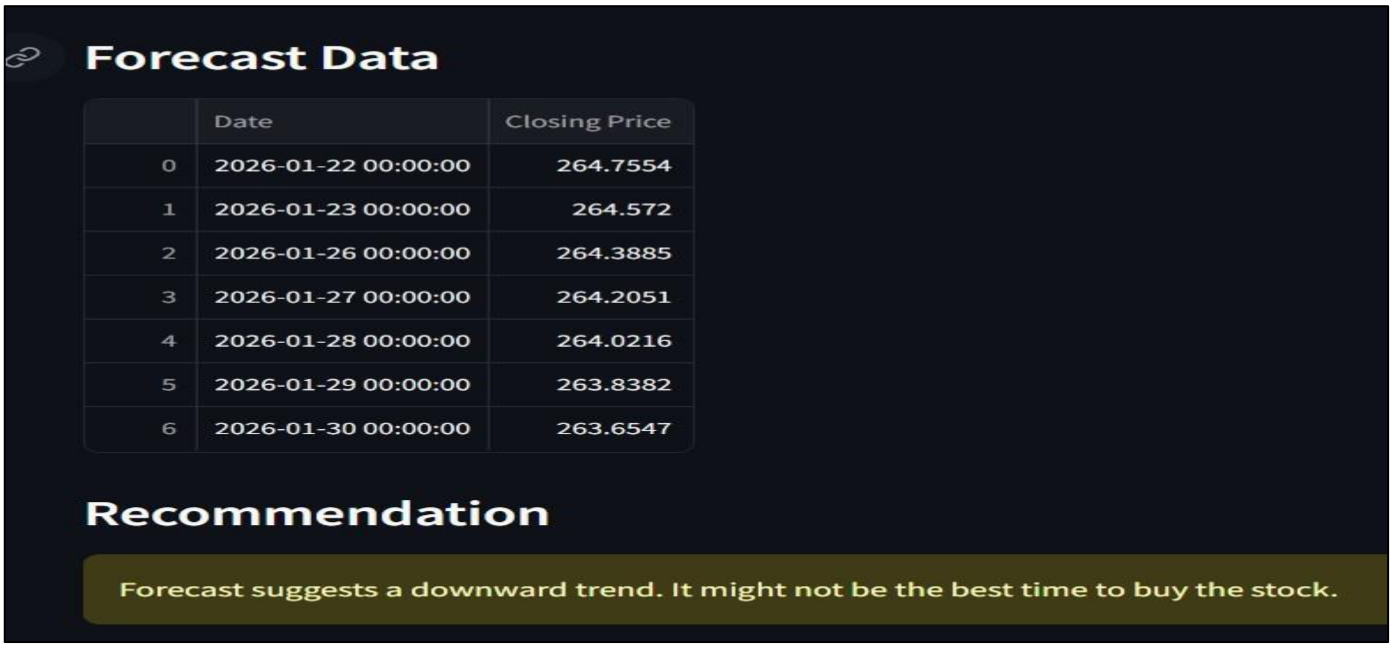
- First, the strategy does not explicitly incorporate measures of volatility, risk-adjusted return metrics (e.g., Sharpe ratio), or drawdown analysis. As a result, the recommendation reflects directional trend expectations rather than risk-aware optimization.
- Second, the model does not integrate trading volume, momentum indicators, or technical signals that may enhance short-term predictive reliability. The decision rule is derived solely from forecasted closing prices, which limits multidimensional market interpretation.
- Third, transaction costs, slippage, bid–ask spread effects, and liquidity constraints are not modeled in the current framework. Consequently, the recommendation output should not be interpreted as an assessment of trading profitability under realistic market execution conditions. Furthermore, the reliability of the recommendation mechanism depends entirely on the predictive accuracy of the underlying forecasting model.
- The model may underperform during earnings announcements, macroeconomic policy changes, or sudden market shocks where price dynamics deviate sharply from recent trends.
- Finally, the recommendation mechanism is entirely dependent on the forecasting model's accuracy. Any degradation in forecast performance—particularly during periods of high volatility, macroeconomic shocks, or earnings announcements—directly affects recommendation reliability.

These limitations are explicitly acknowledged to prevent over-reliance on the system and to reinforce its intended role as a decision support tool rather than an automated trading solution.

➢ *Future Recommendation Enhancements*

Future work may extend the recommendation module by:

- Introducing confidence thresholds
- Incorporating volatility-aware rules
- Adding probabilistic Buy/Hold scores
- Supporting portfolio-level recommendations



Forecast Data

	Date	Closing Price
0	2026-01-22 00:00:00	264.7554
1	2026-01-23 00:00:00	264.572
2	2026-01-26 00:00:00	264.3885
3	2026-01-27 00:00:00	264.2051
4	2026-01-28 00:00:00	264.0216
5	2026-01-29 00:00:00	263.8382
6	2026-01-30 00:00:00	263.6547

Recommendation

Forecast suggests a downward trend. It might not be the best time to buy the stock.

Fig 5 Forecast Data AND Recommendation

VII. LIMITATION AND ETHICAL CONSIDERATION

➢ *Technical Limitations*

- Inability to model nonlinear price movements
- The system does not implement probabilistic Bayesian uncertainty modeling.
- No incorporation of macroeconomic or sentiment data

➢ *Ethical Considerations*

Financial forecasting systems carry ethical responsibility. The application clearly states that:

- Forecasts are probabilistic
- Recommendations are informational, not financial advice
- Users bear responsibility for investment decisions

VIII. CONCLUSION AND FUTURE WORK

This paper presented StockBuddy Assistant, a web-based decision support platform for stock analysis and short-term forecasting. The system demonstrates how interpretable machine learning models can be integrated into accessible financial applications while maintaining low latency and deployment efficiency. By combining rolling-window forecasting, baseline comparison, and uncertainty-aware visualization, the proposed framework provides a transparent and practical tool for exploratory financial decision support, consistent with modern decision support system design principles [8], [9].

While the Linear Regression approach performs effectively in capturing short-term monotonic trends, its reliance on linear assumptions limits performance during periods of heightened volatility and structural market shifts commonly observed in financial markets [2]. The recommendation module is intentionally conservative and designed for informational purposes rather than trading automation.

➢ *Future Work Will Focus on:*

- Incorporating advanced forecasting models such as ARIMA [1], Prophet, and LSTM architectures [3] under controlled deployment constraints.
- Investigating compressed or knowledge-distilled deep learning models to evaluate whether accuracy improvements can be achieved without sacrificing interpretability or inference latency [7].
- Integrating volatility-aware and risk-adjusted recommendation mechanisms to address limitations of linear forecasting under non-stationary conditions [2].
- Extending evaluation through formal statistical hypothesis testing and robustness analysis.
- Supporting multi-asset and portfolio-level analytics for broader financial decision support.

Overall, this work highlights the importance of balancing predictive performance, interpretability, and deployment feasibility in real-time financial decision-support systems [8], [9].

REFERENCES

[1]. R. H. Shumway and D. S. Stoffer, Time Series Analysis and Its Applications: With R Examples, 4th ed. New York, NY, USA: Springer, 2017.
[2]. T. Bollerslev, “Generalized autoregressive conditional heteroskedasticity,” Journal of Econometrics, vol. 31, no. 3, pp. 307–327, 1986.
[3]. I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Cambridge, MA, USA: MIT Press, 2016.
[4]. Alpha Vantage, “Stock Market APIs.” [Online]. Available: https://www.alphavantage.co/

[5]. T. Singh, S. Choudhary, and P. Kumar, "An efficient real-time stock prediction exploiting incremental and offline–online learning," Scientific Reports, vol. 12, no. 1, 2022.
[6]. S.-J. Yu, "The Design of an Intelligent Lightweight Stock Trading System," Applied Sciences, vol. 13, no. 4, 2023.
[7]. D. Campos, M. W. R. da Silva, and J. Gama, "LightTS: Lightweight Time Series Classification with Knowledge Distillation," in Proceedings of the ACM Conference on Information and Knowledge Management (CIKM), 2023.
[8]. H. Subramanian, A. Gupta, and R. Bansal, "A decision support system using signals from social media for stock market analysis," Decision Support Systems, vol. 173, 2024.
[9]. G. Kostopoulos, E. P. Kastritis, and P. Pintelas, "Explainable Artificial Intelligence-Based Decision Support Systems: A Survey," Electronics, vol. 13, no. 2, 2024.