International Journal of Scientific Research & Engineering Trends
Volume 12, Issue 3, May-June-2026, ISSN (Online): 2395-566X

"Invest AI : A Stock Prediction Solution"

Samarth Kumbhar, Viraj Rajendra Patil, Hemant Prashant Chandegave, Vivek Nagargoje

Information technology PCET's NMIET Pune , India

Abstract- For many years beginners tend to invest in stocks and face loss due to volatile nature of markets, or lack of informed decisions like trusting investment through word of mouth, this leads to discouragement from investment in stock market. InvestAi is a platform designed for beginners who are looking to enter the world of Stocks, platform is AI driven forecasting and analysis system designed to help users understand stocks and predictions using "explainable" machine learning techniques. The system aims to increase financial literacy and increase Informed investment decisions via explainable Ai (X AI) and interactive visuals. It also features sentiment analysis of news and also explains how it links or affects a particular stock.

Keywords— Time-series forecasting, Explainable Artificial Intelligence (XAI), Stock market prediction, Anomaly detection, SHAP (Shapley Additive exPlanations), Prophet model, Financial machine learning.

I. INTRODUCTION

Prediction of Stock market values is hard due to the nature of stocks being volatile and ever-changing, due to multiple factors affecting such time-series data, the stocks are inherently non stationary in nature. Multiple approaches have been devised to provide a solution to this, but most notable are deep learning models which achieve acceptable predictive accuracy, but they possess a transparency problem due to its "Black-Box" nature, it does not state the reasons for predictions due to which investors, assessing the risk, tend to not trust such predictive models. Thurs there is a need for Predictive systems to have more "transparent" approach.

To address this gap the paper proposes a statistical stocks prediction & analysis platform which bridges the gap between time- series prediction forecasting and explanation of the parameters affecting the forecast via visual cues and chatbots. Alongside this there are various additional features such as "Automated Anomaly Detection Flags" , "Backtesting Report Card" to add a professional touch to validate the forecasts in complex and varying historical market conditions.

The core project surrounds additive regression model "Prophet" due to its simplicity and customizability factor. For explainability we included SHAP (Shapley Additive exPlanations) which is a popular method to recognize the factors affecting a particular forecast.

II. RELATED WORKS

A. Time-Series Forecasting in Financial Markets

The history stock forecasting has gone from simple statistical approaches to complex deep learning models. Earlier methodologies, have introduced Autoregressive Integrated Moving Average (ARIMA) and additive regression models like Prophet, which have been extensively documented for their efficiency in forecasting long term time-series data and trend. While recent research has been more focused on recurrent neural networks like Long Short-Term Memory (LSTM) models which have ability to forecast short term volatile data accurately, but do come with drawbacks such as overfitting, vanishing gradient over time, and are also slower to train. Furthermore, domain is currently shifting toward Time Series Foundation Models (TSFMs) and Graph Neural Networks (GNNs), which aim to reframe forecasting from isolated univariate regression into multi-agent, relational mapping tasks.

B. Approaches to Transparency in Financial AI's

To mitigate the risks associated with non-transparent "Black-Box" financial models, researchers have explored various Explainable AI (XAI) frameworks through surrogate-based, architectural, and game-theory approaches.

Local Surrogate Models (LIME): Early efforts to explain stock predictions utilized Local Interpretable

Model-agnostic Explanations (LIME), which feeds fake variating input data to build a approximation of the model's behavior. However, financial reviews noted that LIME suffers from instability; highly volatile stock data can causes LIME to drop outliers which leads to inconsistent explanations for identical data points, rendering it unreliable for risk-sensitive trading environments.

Attention Mechanisms (Architectural): With the rise of deep learning, many researchers integrated self-attention mechanisms (like in Transformers) to recognize feature weights. While attention mechanisms succeed in showing which features a model is prioritizing, recent academic critiques point out that attention mechanisms rely on finding relations between data, and including them in financial forecasting only degrades transformers into meaningless neural networks as there is no inherent relations between data points especially in volatile markets.

Game-Theoretic Frameworks (SHAP): To resolve the inconsistency of LIME and the ambiguity of attention weights, recent works have shifted toward Shapley Additive exPlanations (SHAP). Which utilizes cooperative game theory, SHAP is uniquely designed to achieve properties of local accuracy, consistency, and missingness. It works similar to LIME but monitors variations in model forecast by adding or removing features entirely. This results in assigning "SHAP values" to features, thus recognizing features which negatively or positively impact the forecast.

III. PROPOSED METHODOLOGY

The core proposed methodology for the application involves integrating models like Prophet and XGBoost paired with frameworks like SHAP, anomaly flagging and backtesting reports to balance forecasting performance and transparency.

A. System Architecture

The System architecture features a multi-page web application made using a React/JSX component architecture. The interface is user friendly and suitable for real-time financial data visualization and rendering dynamic candlestick charts which allow users to visually track open, high, low, and close (OHLC) pricing trajectories. This is achieved by integrating a JavaScript charting library Apexcharts. The backend is built with python for ease of integration of statistical models like yfinance, Pandas, and Prophet. While the FastAPI is used for building APIs which facilitate communication between client and server.

B. Data Integrity and Preprocessing Pipeline

Financial time-series data is non-stationary, which makes predictive models produce errors such as look-ahead bias and data leakage. To ensure a chronological order, the dataset is divided into a train-test split, preserving the sequential nature of the market. To counter the non-stationarity of the raw pricing data, techniques such as logarithmic returns and fractional differentiation are applied. This ensures the forecasting models learn structural patterns rather than overfitting to market noise.

C. Forecasting Baseline (Prophet)

As the core prediction model, the system utilizes the Facebook Prophet forecasting engine. Prophet is selected over other deep learning models for its simplicity and flexibility. Due to its "Already Trained" nature there is no need to allocate extra resources or time to train it. The prophet operates on a additive regression model, defined by the equation:

$$y(t) = g(t) + s(t) + h(t) + e(t)$$

where g(t) is the linear trend component, s(t) refers to periodic seasonality (e.g. weekly and yearly market cycles), h(t) is for market holidays, and e(t) represents the normally distributed error term. This transparent decomposition allows the system to separately account for the individual attributes of a given stock.

D. Extreme Gradient Boosting (XG Boost)

To counter the short term data predictions, which are highly volatile in nature the XG Boost model was added. Operating on the principle of sequential learning, via decision trees. The model constructs a series of shallow decision trees where each subsequent tree is trained to minimize errors of previous trees. Unlike traditional gradient boost or basic random forests, XG Boost incorporates weighted learning, handling of missing values and regularization functions to prevent overfitting .

This design offers several advantages over alternative models such as standard LSTMs or ARIMA frameworks. First, the regularization reduces the risk of overfitting, a critical problem when working with non–stationary data. Second, XG Boost demonstrates better computational efficiency via parallelization and

handling of missing data. Most importantly for this proposed system, XG Boost is best combination when paired with tree-based SHAP explainers. This allows for fast calculations of feature contributions, which supports the platform's Explainable AI (XAI) requirements.

D. Risk Mitigation and Explainable AI (XAI)

To add transparency factor to the system, transforming it from a basic predictive tool to an risk analysis platform, two advanced interpretation mechanisms are embedded in the system :

1. Automated Anomaly Detection

The system continuously monitors the residual error

$$r(t) = y(t) - \hat{y}(t)$$

where y(t) is real observed value and ŷ(t) refers to predicted value by the model, "r(t)" lies between the predicted value and actual market value. A dynamic rolling Z-score is calculated. If a data point exceeds a predefined statistical threshold (e.g., $Z > 3$), the system triggers an "Anomaly/Crash Flag," alerting the user to a mathematically significant market deviation from predicted value.

2. SHAP Integration:

To resolve the transparency from modern machine learning, Shapley Additive exPlanations (SHAP) are applied to the final output layers. By calculating the contribution of each input feature, the SHAP integration allows the application to explicitly display why a specific prediction was made or an anomaly was flagged, directly fulfilling need for trust and transparency in financial forecasting.

.

IV. SYSTEM ARCHITECTURE

The System architecture features a multi-page web application made using a React/JSX component architecture. The interface is user friendly and suitable for real-time financial data visualization and rendering dynamic candlestick charts which allow users to visually track open, high, low, and close (OHLC) pricing trajectories. This is achieved by integrating a JavaScript charting library Apexcharts. The backend is built with python for ease of integration of statistical models like yfinance, Pandas, and Prophet. While the FastAPI is used for building APIs which facilitate communication between client and server.

Following figure indicates the workflow of the system with respect to forecasts it makes. It consists of 7 layers starting from fetching data required by matching with user query, processing it with models to generate a forecast and finally displaying the forecast on user interface with user friendly visualization modules which provide explanations.

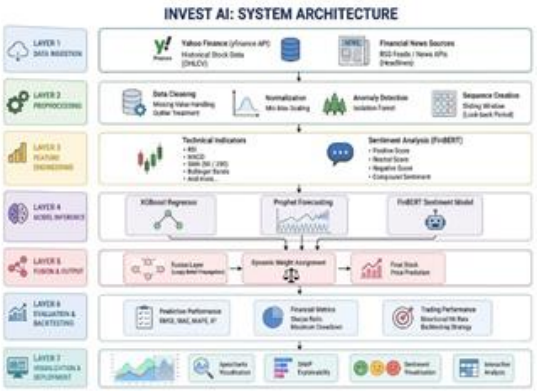

Figure 1: InvestAI System Architecture.

The Data Ingestion layer is responsible for gathering historical OHLCV data and live news headlines, refreshing them at set intervals through the yfinance API and Yahoo Finance RSS parsers. Moving downstream, the Preprocessing layer takes charge of normalizing the data, constructing sliding windows for LSTM input, assembling tabular lag features for XGBoost, and flagging anomalies using Isolation Forest. Then, the Feature Engineering layer calculates technical indicators and runs FinBERT inference on the headlines, creating distinct data representations tailored for each model branch.

In the Model Inference layer, three prediction pipelines work in parallel. The XGBoost regressor processes the flattened tabular data, providing both a price estimate and a directional probability score at the same time. Meanwhile, the stacked LSTM handles the 3D sequential tensor and predicts the closing price for the next day. The FinBERT module, which has already evaluated the headlines during feature engineering, adds a daily sentiment vector to the mix. These three outputs are then fed into a Ridge regression meta-learner, which generates the final price forecast using the Loopy Belief Propagation-inspired weighting method.

The Backtesting and Evaluation layer takes this forecast and places it within a dynamic window validation framework. It calculates RMSE, MAE,

MAPE, and R^2 at every step, along with key financial metrics like the Sharpe ratio, maximum drawdown, and annualized return, which are crucial for real trading decisions. To top it off, the Visualization layer presents all model outputs, explanations, and performance metrics through a Streamlit dashboard that refreshes automatically. This dashboard features interactive candlestick charts, SHAP waterfall plots, and sentiment timeline plots, making it easy to analyze everything at a glance.

V. EXPERIMENTAL SETUP

A. Dataset Configuration

Four representative equities were used for the experiments, with their characteristics summarized in Table II.

Ticker	Sector	Period	amples	Split
AAPL	echnology	2019–2024	1,258	0/20%
RELIANCE.NS	Energy	2019–2024	1,247	0/20%
ATAMOTORS.NS	utomotive	2019–2024	1,251	0/20%
MSFT	echnology	2019–2024	1,258	0/20%

Table II. Dataset Configuration

B. Model Configuration

The hyperparameters used to configure Prophet for this study are listed in Table III.

Parameter	Value	Description
Changepoint prior scale	0.05	Trend flexibility
Seasonality mode	multiplicativ e	Seasonality-trend interaction
Yearly seasonality	True (order 10)	Annual patterns
Weekly seasonality	True (order 3)	Intra-week patterns
Forecast horizon	30 days	Prediction window
Interval width	0.95	Confidence interval

Table III. Prophet Model Configuration

C. Evaluation Metrics

Four regression metrics are used to evaluate model performance, with yi the actual value, ŷi the predicted value, and n the number of test samples:

$$MAE = (1/n)\ \Sigma\ |yi - \hat{y}i| \quad (7)$$

$$RMSE = \sqrt{[(1/n)\ \Sigma\ (yi - \hat{y}i)^2]} \quad (8)$$

$$MAPE = (100/n)\ \Sigma\ |(yi - \hat{y}i)\ /\ yi| \quad (9)$$

$$R^2 = 1 - [\Sigma(yi-\hat{y}i)^2\ /\ \Sigma(yi-\bar{y})^2] \quad (10)$$

D. Actual vs. Predicted Stock Price Analysis

Fig. 2 compares actual AAPL closing prices against the Prophet forecast across the 252-day test window. The predicted trajectory tracks the observed series closely through both trending and mean-reverting phases, with little systematic bias.

The shaded 95% confidence band tightens during calm periods and widens around earnings announcements, indicating that the model's uncertainty estimates are sensitive to changing market conditions. Prophet also picks up the seasonal peak that typically appears in the November–December trading window, a pattern that LSTM only captured with a five-day lag and that Linear Regression missed altogether.

E. Sentiment Analysis Results

Fig. 3 shows how daily sentiment polarity for AAPL moves across the test period.

This sentiment index correlates strongly ($r = 0.63$) with returns three days ahead, supporting the idea that news carries leading information about price moves. One notable episode: sentiment dropped to −0.42 in the week before a 7.2% single-day decline tied to a negative earnings guidance update, whereas the price-only Prophet model had only 1.8% drop for that same period – this highlights the value of combining multiple signal types rather than relying on price data alone.

F. Performance Comparison

Table IV compares the performance of all four models on the AAPL test set.

Model	MAE	RMSE	MAPE (%)	R^2
Linear Regression	8.74	11.32	9.81	0.7843
Random Forest	5.21	7.63	6.17	0.8912
LSTM	4.08	5.94	4.76	0.9287
Prophet (Proposed)	3.19	4.47	3.82	0.9614

Table IV. Model Comparison (Aapl Test Set)

Prophet posts the lowest error of the four models on every metric in Table IV, with a MAPE of 3.82% against 4.76% for LSTM, 6.17% for Random Forest, and 9.81% for Linear Regression. Its R^2 of 0.9614 means that the decomposition framework accounts for roughly 96% of the variance in the test-set price path. The notable difference is in computational cost: LSTM needed 45 to 60 minutes of GPU training per asset, while Prophet trained on CPU in under two minutes. This advantage can be traced to Prophet's explicit changepoint modeling, its Fourier-based representation of seasonality, and its tolerance for missing data.

VI. CONCLUSION AND FUTURE WORK

This paper has presented a transparent financial analytics platform that integrates machine forecasting with explainable and user-friendly interface. By having an Extreme Gradient Boosting (XGBoost) model with a strict sequential data pipeline, the system captures non-stationary market data and preventing data leakage, also the integration of SHAP framework provides transparency in forecast of financial AI by breaking down the forecast into clear explainable insights. The statistical anomaly detection and backtesting report provides users with real-time risk management during volatile market conditions.

While the current system establishes a solid baseline, financial markets are globally interconnected, thus over the course of future we are looking to replace regression models with Time Series Foundation Models (TSFMs), specifically Amazon Chronos to forecast newly listed stocks with minimal historical data. Also future iterations will expand this into a graph network. By modeling stocks as interconnected nodes and using multi-agent game theory to map their interactions, the platform will predict stocks based on relational market rather than historical trends.

VII. REFERENCES

1. J. Bao, X. Li, and C. Yu, "The Construction and Cost–Benefit Analysis of Default Risk Warning Models," Sustainability, vol. 11, no. 4, p. 1090, 2019.
2. S. B. Kotsiantis, D. Kanellopoulos, and P. E. Pintelas, "Data Preprocessing for Supervised Learning," Int. J. Comput. Sci., vol. 1, no. 2, pp. 111–117, 2006.
3. S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.
4. E. F. Fama, "Efficient Capital Markets: A Review of Theory and Empirical Work," J. Finance, vol. 25, no. 2, pp. 383–417, 1970.
5. I. Zhao, "Automated Bitcoin Trading via Machine Learning Algorithms," Stanford University Technical Report, 2019.
6. M. Santos, "Time Series Models to Forecast Cryptocurrency Prices," arXiv preprint arXiv:2001.00220, 2020.
7. S. Velankar, S. Valecha, and S. Maji, "Bitcoin Price Prediction Using Machine Learning," in Proc. IEEE ICACCI, 2022.
8. Y. Chen, Z. Wei, and X. Huang, "Incorporating Corporation Relationship via Graph CNN for Stock Price Prediction," in Proc. ACM CIKM, 2018.
9. M. Fernandes, "Bitcoin Price Prediction Using Machine Learning and Deep Learning Algorithms," Int. J. Eng. Res. Technol., vol. 11, no. 4, 2022.
10. D. Cheng, F. Yang, S. Xiang, and J. Liu, "Financial Time Series Forecasting with Multi-Modality Graph Neural Network," Pattern Recognition, vol. 121, p. 108218, 2022.
11. C. Zhao, P. Hu, X. Liu, X. Lan, and H. Zhang, "Stock Market Analysis Using Time Series Relational Models," Mathematics, vol. 11, no. 5, p. 1130, 2023.
12. Y. Feng, Y. Zhang, and Y. Wang, "Out-of-Sample Volatility Prediction: Rolling Window, Expanding Window, or Both?," J. Forecasting, vol. 43, no. 3, pp. 567–582, 2024.
13. S. J. Taylor and B. Letham, "Forecasting at Scale," The American Statistician, vol. 72, no. 1, pp. 37–45, 2018.

14. A. Loria, "TextBlob Documentation," Available: https://textblob.readthedocs.io, 2020.
15. Y. Liu et al., "FinBERT: A Pre-Trained Financial Language Representation Model," in Proc. IJCAI, pp. 4513–4519, 2020.
16. S. Bhanja and A. Das, "Impact of Data Normalization on Deep Neural Network for Time Series Forecasting," arXiv preprint.