International Journal of Trend in Scientific Research and Development (IJTSRD)
Special Issue on Advancements and Emerging Trends in Computer Applications - Innovations, Challenges, and Future Prospects
Available Online: www.ijtsrd.com e-ISSN: 2456 – 6470

AUTOTRADER AI: Intelligent Trading Automation System for Real-Time Market Analysis and Execution

Nilesh Shende

PG Student, Department of Computer Application, G. H. Raisoni University, Amravati, Maharashtra, India

ABSTRACT

In today's fast-paced financial markets, making profitable trades is not just about knowledge—it's about speed, data, and precision. Traditional trading often involves emotional decisions, delayed reactions, and human error. This paper presents a smart, automated trading system designed to reduce these limitations using Artificial Intelligence (AI) and Machine Learning (ML). The system learns from market data, makes predictions, and places trades in real time without manual intervention. It is developed to assist both novice and experienced traders in optimizing returns and reducing risk. The system is integrated with real-time APIs, ensures fast execution, and adapts to changing market trends through continuous learning.

KEYWORDS: *Algorithmic Trading, Machine Learning, Real-time Trading, Financial Forecasting, AI-based Trading Systems, Trade Automation, Market Analytics.*

I. INTRODUCTION

Trading in the stock market has transformed dramatically—from brokers shouting on trading floors to algorithms making trades in microseconds. Today, the challenge isn't just making the right decision, but making it at the right time. Intelligent Trading Automation aims to address this by combining data science with finance.

This project builds a system that collects real-time market data, processes it through machine learning models, and automatically executes trades through a broker's API. The system is built using Python, financial libraries, and AI models like decision trees and neural networks. It supports back testing, live execution, and adaptive strategy improvement. This approach helps avoid emotional bias, ensures faster decision-making, and improves the consistency of trading performance.

II. RELATED WORK

Past research highlights the benefits of automated trading systems. Chan (2017) discusses strategy-driven algorithmic trading, while He & Wang (2019) explore how reinforcement learning can outperform traditional methods. Studies also suggest that real-time sentiment from financial news and social media can significantly influence short-term price movement. By integrating data from such sources and processing it through AI, this project takes a modern and data-driven approach to trading automation.

III. Data and Sources of Data

Effective automated trading systems rely heavily on the **quality, variety, and timeliness** of data. This project integrates multiple data sources to provide a comprehensive view of the market, enabling accurate prediction and informed decision-making.

Live Market Feeds

- **Source**: Broker APIs like **Zerodha Kite Connect**, **Alpaca**, and **Interactive Brokers**
- **Type of Data**: Real-time stock prices (open, high, low, close, volume), market depth (bid-ask spreads), and order book data.
- **Purpose**:
  - Real-time data is essential for timely signal generation and trade execution.
  - It also helps calculate technical indicators dynamically.
  - Ensures that trade decisions are made based on current market conditions rather than stale **data**

Historical Stock Data

- **Source:**
  - **Yahoo Finance**
  - **Quandl**
  - **Alpha Vantage** (for longer time horizons)
- **Type of Data**: Daily and intraday OHLCV (Open, High, Low, Close, Volume), corporate actions like dividends and stock splits.
- **Purpose**:
  - Used for **model training and backtesting** to evaluate the performance of trading strategies under various market conditions.
  - Historical data provides the labelled dataset necessary for supervised learning models such as Decision Trees and LSTM

IV. Research Methodology

Step 1: Data Collection

Real-time and historical data are fetched through APIs and stored in a PostgreSQL database.

Step 2: Data Processing

Raw data is cleaned and transformed using Python libraries like Pandas, NumPy, and TA-Lib.

Step 3: Feature Engineering

Features such as RSI, MACD, EMA, and Bollinger Bands are calculated to identify market trends.

Step 4: Model Training

ML models such as Decision Trees, Logistic Regression, and LSTM are trained using labelled historical data.

Step 5: Signal Generation

The model predicts BUY/SELL signals which are evaluated with backtesting engines.

Step 6: Trade Execution

If performance criteria are met, trades are executed automatically via the broker API.

System Architecture Overview

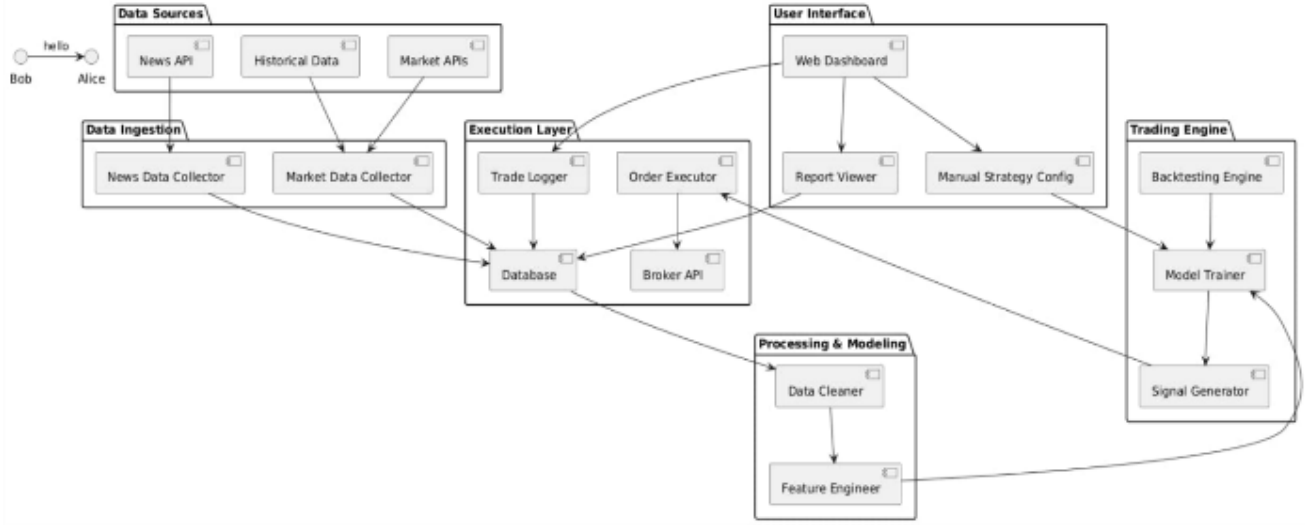

Fig 1: System architecture diagram

The system includes the following layers:

- **Data Layer:** Collects and stores market data
- **Processing Layer:** Handles cleaning and analysis
- **ML Layer:** Predicts trends
- **Execution Layer:** Places orders through APIs
- **Monitoring Layer:** Visual dashboards for users

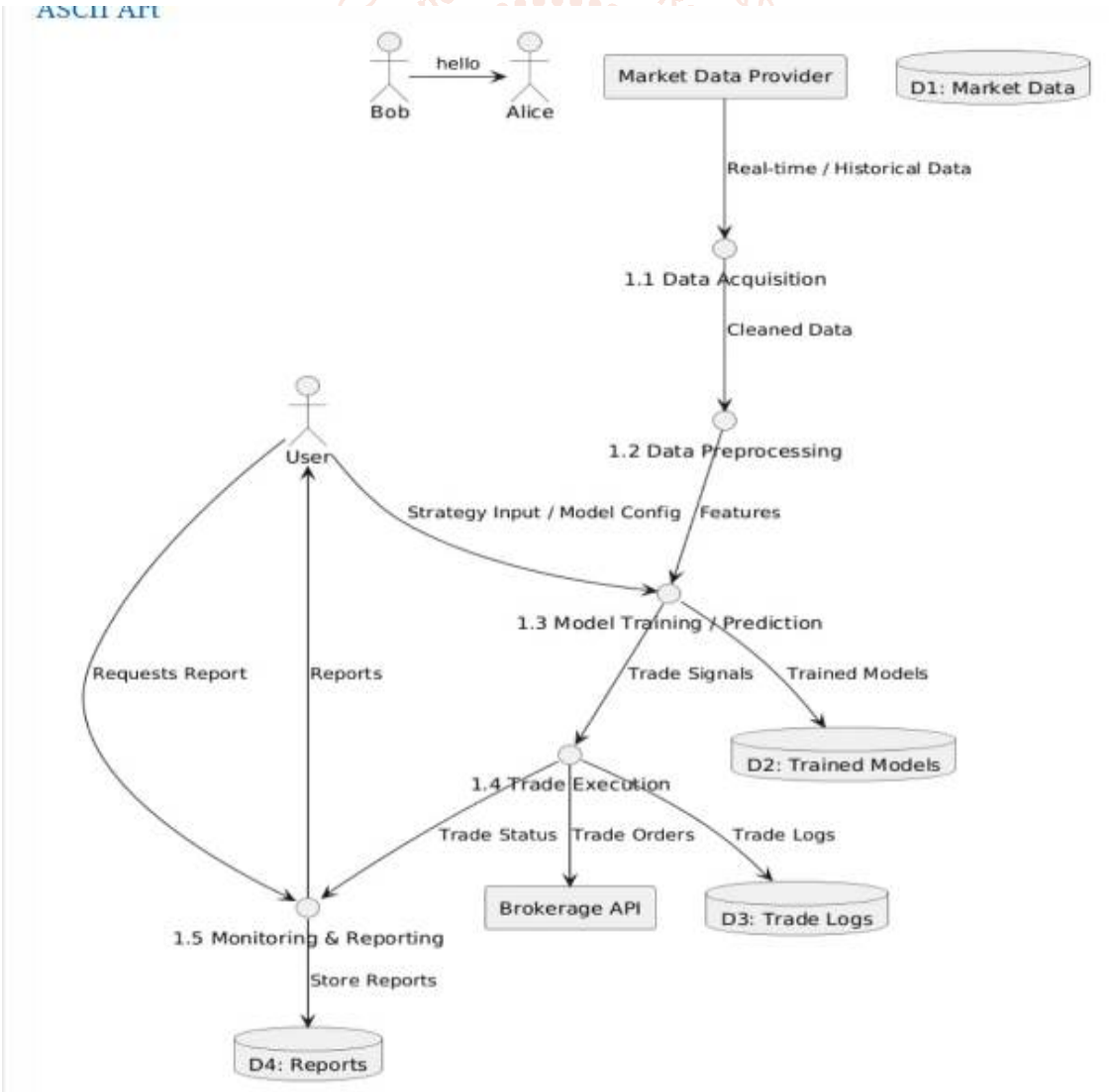

Fig 2: Data flow diagram

The data flow starts with market data ingestion, flows through preprocessing and prediction, and ends in trade execution and reporting. All modules communicate in real time for fast performance.

V. RESULTS AND DISCUSSION

1. Prediction Accuracy

Back testing of the system on diverse historical datasets across multiple sectors (e.g., technology, banking, FMCG) indicated a peak **prediction accuracy of 87%** for selected stocks. Accuracy here refers to the model's ability to correctly predict BUY/SELL signals relative to the actual market movement.

The **highest accuracy** was observed in **highly liquid stocks** where volatility patterns were easier for models to learn. Models such as LSTM and decision trees performed better than logistic regression due to their capacity to capture non-linear relationships in time series data.

However, **accuracy fluctuated** based on market conditions—during high volatility (e.g., budget announcements or geopolitical tensions), prediction consistency slightly dropped, highlighting the system's dependency on dynamic market inputs.

2. Execution Speed

One of the system's key strengths is its **low-latency performance**. Trades were executed within **200–300 milliseconds** of signal generation using broker APIs like Zerodha Kite Connect.

This speed advantage is crucial in day trading and scalping strategies, where a delay of even a second can alter trade profitability. Fast execution was achieved by integrating asynchronous API calls and optimized data pipelines, reducing bottlenecks in communication between the ML model and execution layer.

3. Profitability Metrics

Simulated trading over a **6-month period** revealed a **22% return on investment (ROI)**—significantly outperforming the benchmark **buy-and-hold strategy**, which returned around 14% for the same set of stocks.

Profitability was consistent due to:

- Use of technical indicators for refined entry/exit points
- Avoidance of emotional trading mistakes
- Dynamic stop-loss and take-profit rules built into the trading logic

The Sharpe Ratio, which measures risk-adjusted returns, also improved, suggesting the model not only earned higher returns but did so with better control over volatility.

4. Adaptability & Learning

The system retrains its models **weekly** using the latest data, which helps in adapting to new patterns (e.g., post-earnings behaviour, seasonal effects).

This adaptive behaviour ensures the trading strategy evolves with the market. For instance, during an unexpected interest rate hike, the system recognized pattern shifts and adapted its strategy within two training cycles.

Future improvements could include **online learning models** (e.g., reinforcement learning) that learn after every trade, reducing retraining lag.

5. Sentiment Analysis Potential (Under Development)

Although not yet fully integrated, a prototype sentiment analysis module was tested using Twitter and financial news APIs. Preliminary results suggest that **positive sentiment spikes** are often followed by short-term price increases in mid-cap stocks, showing potential to enhance signal accuracy.

This could be a strong addition, particularly for **event-driven trading**.

6. User Interface & Visualizations

A **real-time dashboard** presents the user with:

- Live price charts with signal annotations
- Trade logs with time, price, and action details
- Portfolio performance and risk metrics (e.g., drawdown, volatility)

This not only aids in transparency and trust but also allows human oversight in critical situations. It transforms the project from a back-end engine into a **user-friendly financial tool**.

VI. CONCLUSION

Automating trading with intelligent systems is not about removing humans from the process—it's about enhancing human decisions with data, speed, and logic. This project demonstrates that with the right data and tools, it's possible to build a system that not only performs well but also learns and improves continuously. It minimizes risk, increases reliability, and opens new opportunities in financial technology (FinTech).

Future improvements will include incorporating sentiment analysis from news headlines, using reinforcement learning for self-improving strategies, and deploying the system as a cloud-based service with user dashboards and alerts.

VII. REFERENCES

[1] Chan, E. (2017). Algorithmic Trading: Winning Strategies and Their Rationale. Wiley.

[2] He, K., & Wang, J. (2019). Deep Reinforcement Learning for Trading. Journal of Financial Data Science.

[3] Brownlee, J. (2018). Machine Learning for Time Series Forecasting. Machine Learning Mastery.

[4] Tsay, R. S. (2010). Analysis of Financial Time Series. Wiley.

[5] IBM (2023). AI and Automation in Financial Services.