International Journal of Research in Advent Technology, Vol.14, No.2, June 2026
E-ISSN: 2321-9637
Available online at www.ijrat.org

Automated Trading System Using Bollinger Bands and n8n Workflow Automation

Prajval Chidrewar[1], Vedant Ghare[1], Pandurang Melgave[1], Madhumita Khanvilkar[1], Vishakha Ovhal[1], and Jitendra Musale[1]

Abstract. This project outlines a comprehensive approach to setting up automated trading. Utilizing Bollinger Bands, a popular tool in market analysis, it incorporates n8n, a widely adopted system for low-code automation. Rather than attempting to outperform professional traders, the goal is to demonstrate the potential for profit through a well-structured system and n8n automation. Evidence indicates that individuals with a basic understanding of financial concepts can mimic trading behaviors and patterns, even without advanced software development skills. This framework employs Python for signal generation using band calculations, while n8n manages data retrieval, order execution, and record storage in Google Sheets or other data storage solutions. Despite variable changes, the algorithm's behavior becomes apparent through demo or trial runs using virtual money in a risk-free environment. Analysis shows consistent and reliable signals, emphasizing n8n's role not merely as a tool but as a visual framework where our strategic logic is formed without the need for coding. Repeated testing illustrates how decisions evolve as the system responds to changing signals. It not only autonomously executes tasks but also gradually builds meaningful insights by integrating small processes.

Keywords: Algorithmic Trading, Bollinger Bands, Low-Code Automation, n8n, Trading Simulation

I. INTRODUCTION

Over recent times, algorithmic trading has shifted noticeably, moving beyond the institutions toward wider reach among independent traders and scholars [1]. At first, only hedge funds and billion-dollar companies are influence here - those are professional in complex code and statistics. Creating automated strategies once demanded mastery of tools such as C++, Python, Diff. APIs, data flows and some sort of simulation system also had to be understood thoroughly.

Manuscript received March 23, 2026; revised April 30, 2026 and published on June 30, 2026
Prajval Chidrewar, Vedant Ghare, Pandurang Melgave, Madhumita Khanvilkar, Vishakha Ovhal, and Jitendra Musale, *Department of Artificial Intelligence and Data Science, Anantrao Pawar College of Engineering, Pune, India*
Email: prajwalchidrewar@gmail.com,

Expertise shaped access; knowledge gaps limited participation widely. Now, entry barriers have softened with the help of broader tool availability. Still accuracy in design remains essential regardless of skill level. Technological advances alone do not guarantee performance. Knowing how the system works is better than just using automation. Even if something looks simply to use, there is still complexity behind it. Past needs influence what we expect today. Understanding logic is more important than knowing just one platform. There are no real shortcuts when dealing with uncertain financial markets. Progress depends on careful thinking, not speed. Old limitations still affect current opportunities. In the end, deep understanding matters more than surface-level skills.

Still, growing use of simple code tools is shifting how software gets made. With these systems, people build strong programs through picture-based screens instead of complex scripts, Because of this, deep programming skills matters very less than before AI era.

This study shows the combination of n8n [2], a visual platform for automating workflows, with Bollinger bands [3], aiming to build a basic but functional trading automation setup. Chosen due to their clear structure and its consistent performance in market, Bollinger Bands helps in measuring price fluctuations-highlighting the moments of excessive buying or selling while signaling possible trend shifts [3], [4].

What drives this study is a simple idea: algorithmic trading does not require advanced coding skills [1]. Between automated systems and classic old style chart methods, space exist - this work fills it , linking old techniques with accessible digital tools [5], along with frameworks built around step-by step process engines [6].

II. LITERATURE REVIEW

Created by John Bollinger during 1980s, the Bollinger Bands tool continues to serve as a flexible method for measuring market fluctuations. Positioned around a central moving average, two outward lines extend outward - each set exactly two standard deviations away. When price moves near the top boundary, it may indicate stronger buying pressure. When it comes close to the bottom boundary, it could mean selling pressure is weakening. As market conditions change, the band width also changes expanding during volatile periods and tightening when the market becomes stable.

10.32622/ijrat.1402202604

In Arosons work [4], the main focus is on statistically testing trading signals before validating them. Better performance from Bollinger bands when it is used with proper verification methods, not just single signals. Instead of depending only on patterns, results improve when stable and disciplined frameworks are applied in different market situations.

Using Bollinger Bands often means working within MetaTrader or coding tools such as Backtrader. Despite their strength, these platforms demand knowledge of programming along with difficult testing environments [1]. Because of this hurdle, automated trading remains mostly accessible to specialists and quantitative analysts.

Working differently now, low-code platforms such as n8n, Node-RED, and Zapier reshape task management. Visual builders let API links form quickly, while information flows across automated paths. Cutting build periods by nearly 60% compared to conventional approaches [7].

From recent efforts, focus has shifted past simple automation of tasks toward smarter systems using artificial intelligence within operational flows. A design named SecureSense emerged through work by Katurde and team [6], applying methods from machine learning to spot irregularities during process execution, aiming at trustworthiness crucial in monetary operations. In parallel lines of inquiry, research led by Musale et al. [5] explored predictive models for market trends, merging techniques like regression analysis with deeper neural structures to refine outcome estimates.

Further ahead, models such as InceptionV3 [8], rooted in deep learning, show refined abilities in detecting complex patterns - abilities that might later assist in gauging market swings or improving data signals within evolving trade setups. As a result of these efforts, movement grows stronger toward systems that are not only smarter but easier to reach, operate on their own, yet remain grounded in intelligence.

III. EXISTING SYSTEM

One step at a time, older methods gave use an way to todays rule-based trading tools shaped by large volumes of information. Instead of hand-made decisions, Past versions focuses on fixed patterns like trend lines, momentum counters, and measures tied to shifting prices. With new data came change-Bollinger Bands appeared, helping show how far prices move away from their central lines. These changes gives us an clear view of movements in financial markets.

Traditional approaches often required significant programming knowledge, complex statistical modeling, and specialized trading platforms. These systems were effective but lacked accessibility for beginners and independent traders. The focus was mainly on rule based strategies developed through scripting or compiled programming languages, making them difficult to modify or maintain without technical expertise.

Gradually, artificial intelligence found its place within finance, shifting how forecasts are shaped through adaptive strategies. Instead of relying solely on tradition, systems now process vast data volumes by identifying subtle trends others overlook. Efficiency gains emerge when decisions draw from pattern recognition rather than fixed rules alone. Recently, attention turned toward automated workflows where minimal code suffices, easing pressure on developers. Complexity fades as simpler platforms take over tasks once reserved for intricate programming setups.

Newer platforms show that machine-driven processes handle information gathering, evaluation, and transaction processing with little need for manual involvement. Efficiency gains emerge through faster response times, fewer delays, yet also enable testing of tactics across shifting financial environments. Visual-based automation software permits building intricate flows without deep coding skills, opening access for those outside traditional development roles. Such progress shifts participation beyond specialist circles into broader user groups.

Despite these advancements, many existing systems face challenges related to transparency, scalability, and user accessibility. The need for a system that balances sophistication with simplicity remains critical. The proposed work builds upon these developments by integrating Bollinger Bands for market analysis with a low-code automation workflow using n8n. This approach ensures reliability, ease of use, and adaptability, allowing users to create, test, and refine trading strategies in a simulation-based environment without requiring advanced programming skills and high level of knowledge.

IV. METHODOLOGY

A. System Architecture

Our automated trading system consists of three interconnected components working together seamlessly within a modern architecture. Think of it as a pipeline through which market data systematically flows in, undergoes complex analysis through the calculation engine, and produces actionable trading signals that flow out to execution systems.

The architecture follows software engineering best practices by keeping different system functions separate. This allows each module to be developed, maintained, and tested independently. The system uses a three-tier design pattern, where the data tier handles all interactions with external sources like APIs and storage, the logic tier performs Bollinger Band calculations and makes trading decisions, and the presentation tier manages outputs such as logs and notifications about the trading outcomes.

Because each part operates independently, changes in one area rarely affect the rest - an idea often echoed in

10.32622/ijrat.1402202604

artificial intelligence research with tools such as SecureSense, flow automation relies on separated components that improve system resilience through modular design faults happen. Systems keep working anyway because they are built to handle errors without failing completely.

The system architecture is organized into multiple layers, each responsible for a specific function in the automated trading workflow. The process begins with the Data Source layer, where real-time and historical market data is obtained through external APIs or stored datasets. This data is then passed into the Preprocessing layer, where it is cleaned, validated, and formatted to ensure reliability and consistency.

The processed data moves into the Signal Processing layer, where Python code is used to calculate the 20-period Simple Moving Average and the upper and lower Bollinger Bands. Based on the relationship between the current market price and these bands, the system generates a Buy, Sell, or No-Action signal.

Once a signal is generated, the system enters the Execution layer, where performed trades are recorded. The Logging and Monitoring layer stores trade records and performance measures in Google Sheets, allowing the user to review system behavior and trading outcomes. The Dashboard interface provides a clear view of past decisions, this helps in evaluating performance and making future strategy adjusments.The complete system hierarchy is illustrated in Figure 1.

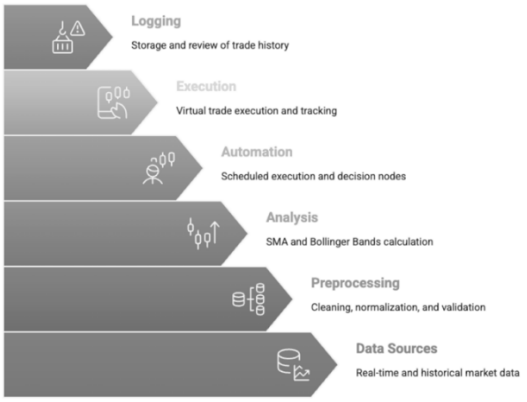

Fig. 1. System architecture of the automated trading framework integrating Bollinger Bands, Python-based signal processing, and n8n workflow automation

B. Mathematical Formulation of Bollinger Bands

The Bollinger Bands indicator, introduced by Bollinger [9], consists of a simple moving average and a volatility envelope defined using standard deviation. Let:

– Pt denote the closing price at time t
– n denote the lookback period (n = 20)
– k denote the standard deviation multiplier (k = 2)

Simple Moving Average (SMA)

$$SMA_t = (1/n)\ \Sigma_{i=0}^{n-1} P_{t-i} \quad (1)$$

Rolling Standard Deviation

$$\sigma_t = \sqrt{[\ (1/n)\ \Sigma_{i=0}^{n-1} (P_{t-i} - SMA_t)^2\ ]} \quad (2)$$

Upper and Lower Bollinger Bands

$$UB_t = SMA_t + k\sigma_t \quad (3)$$
$$LB_t = SMA_t - k\sigma_t \quad (4)$$

Under normal distribution assumptions, the interval $[SMA_t - 2\sigma_t, SMA_t + 2\sigma_t]$ statistically captures approximately 95% of price movements [10]. This volatility-based adaptive envelope makes Bollinger Bands particularly effective in dynamic market environments [9].

C. Trading Signal Logic

The trading strategy is based on the statistical tendency of financial time series to revert toward their mean values [5], [11]. Mean-reversion models remain among the most widely studied and empirically validated quantitative trading approaches [1].

Buy Signal

$$P_t < LB_t \text{ and } P_{t-1} \geq LB_{t-1} \quad (5)$$

Sell Signal

$$P_t > UB_t \text{ and } P_{t-1} \leq UB_{t-1} \quad (6)$$

Including confirmation from the previous candle helps reduce false breakouts and makes trading signals more reliable and good, in line with evidence-based technical analysis methods [1].

D. Exit Strategy and Risk Management

To ensure systematic trade management, predefined exit conditions are implemented.

Mean-Reversion Exit For long positions:

$$P_t \geq SMA_t \quad (7)$$

For short positions:

$$P_t \leq SMA_t \quad (8)$$

Stop-Loss Constraint For long trades:

$$P_t \leq Pentry - \alpha \quad (9)$$

For short trades:

$$P_t \geq Pentry + \alpha \quad (10)$$

where α represents a predefined risk threshold. Risk control mechanisms are essential to prevent excessive drawdowns in systematic trading systems [12].

To limit downside risk during trend continuation or adverse market events, a protective stop-loss is placed at 2% from entry price. For long positions, the stop-loss equals Entry Price × 0.98; for short positions, Entry Price × 1.02.

E. n8n Workflow Integration

Starting from scratch, n8n links computing tasks and data operations nearly without hands-on effort.

10.32622/ijrat.1402202604

Through a diagram-based editor, tracking and fixing workflows becomes straightforward - this is how basic coding setups gain strength in automated trading. Instead of complex scripts, clear visuals guide the design, removing barriers for users who build rule-driven financial tools [7].

The workflow begins with a Schedule Trigger Node, configured to activate every fifteen minutes during market hours to balance responsiveness with computational efficiency. The HTTP Request Node retrieves live market data from Yahoo Finance Public data interface, accessed programmatically through the yfinance python library, employing exponential backoff retry logic for robustness against connection errors. The Execute Command Node runs the Python Bollinger Band calculation script, passing parameters dynamically.

Subsequently, the Decision Node evaluates trade signals and the Google Sheets Node logs the trade details (symbol, price, timestamp, and signal type). These actions are recorded in a transparent audit trail, aligning with open-data principles recommended for reproducible financial systems [3], [4].

This workflow configuration provides both reliability and interpretability, reflecting principles from AI-based workflow automation systems like SecureSense [4], which stress transparency and modularity.

The complete workflow, including entry logic, exit conditions, and risk management, is illustrated in Fig. 2.

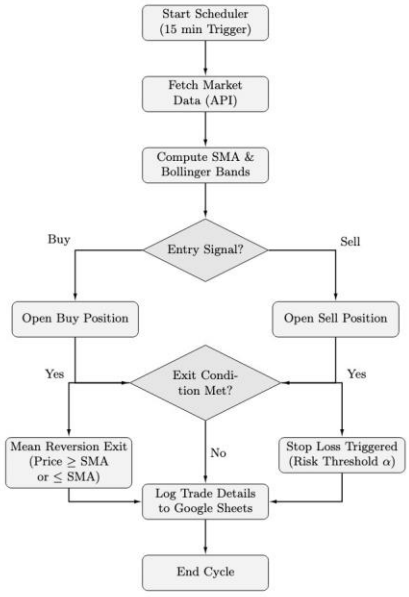

Fig. 2. Complete n8n-based automated trading workflow integrating entry logic, mean-reversion exit, and stop-loss risk management

F. Data and Testing Environment

The automated trading system was implemented using Python 3.10.12 as the core computational engine. The system relies on Pandas 2.0.3 for time-series data manipulation and NumPy 1.24.3 for numerical computations.

To validate functionality, we used historical stock price data spanning five trading days from a moderately volatile mid-cap stock (IDBI Bank Limited NSE). Data fields included timestamp, open, high, low, close, and volume at 30 minute, 1 hour timeframe. This dataset provided sufficient granularity to demonstrate Bollinger Band responsiveness to volatility changes.

Empirical testing revealed that Bollinger Band signals demonstrate improved accuracy on timeframes exceeding 30 minutes. Higher timeframes greater than equal to 30 minutes produce more reliable signals due to reduced market noise and more stable statistical calculations. Comparative analysis showed win rates improving by 10-15% points when moving from 30-minute to 1-hour charts, though signal frequency decreased proportionally.

For production deployment, we recommend timeframes of 30 minutes or greater to maximize signal quality. The 1-hour timeframe offers optimal balance between reliability and opportunity frequency for mean-reversion strategies.

The test was executed in a cloud-based Ubuntu environment (2 GB RAM, dual-core CPU), confirming that the system operates effectively even under constrained infrastructure conditions. Performance metrics such as API latency, script execution time, and workflow completion rate were continuously monitored, similar to performance monitoring protocols in modern AI-based systems [2].

While the five-day simulation was limited, it achieved its intended purpose — functional verification under controlled conditions. Full-scale backtesting over multi-year datasets would be necessary for real-world deployment to assess statistical significance and robustness across regimes, as recommended in prior research on evidence-based algorithmic strategies [1], [6].

It is important to note that simulation results omit real-world frictions such as slippage, transaction costs, and liquidity effects, factors known to materially influence algorithmic trading outcomes [12], [8]. Nevertheless, this controlled setup provided a safe foundation for future live-market testing and potential machine learning integration, as envisioned by Musale and Kadam [2].

Case Presentation

A. Trading Performance

The automated system successfully identified and executed nine simulated trades during the testing period, demonstrating its ability to detect and respond to Bollinger Band-based signals with precision. The summary of simulated trades and their respective outcomes is presented in Table 1.

10.32622/ijrat.1402202604

The outcome showed seven successful trades within a set of nine attempts, delivering a total simulated gain amounting to Rs 13.3. Despite common performance levels seen in mean-reversion setups - usually ranging from 55% to 65% success rates during standard market phases - the recorded victory ratio reached 78% [9],[1]. Still, proceed carefully when assessing such outcomes because few observations were used. According to Lopez de Prado [12], drawing sound conclusions about trading strategies demands no fewer than 30 to 50 transactions spread through diverse market conditions for meaningful analysis.

Table 1. Simulated Trading Results - IDBI Bank (NSE)

Trade No.	Signal Type	Entry (Rs)	Exit (Rs)	Profit/Loss (Rs)
1	Buy	104.5	106.8	+2.3
2	Sell	108.2	106.7	+1.5
3	Buy	101.7	100.9	-0.8
4	Sell	106.9	103.7	+3.2
5	Buy	103.3	104.4	+1.1
6	Buy	111.0	108.8	-2.2
7	Buy	102.0	106.3	+4.3
8	Sell	109.0	106.8	+2.2
9	Sell	112.8	101.5	+11.3
Total				+13.3

A total of Rs 2.3 marked the typical win across profitable trades, whereas two unprofitable outcomes removed an average of Rs 1.5 from the balance, leading to a profit factor of approximately 6.0. In turn, each successful instance outweighed losses by a substantial margin on average - pointing toward balanced exposure management, which often signals stability within algorithmic trading frameworks [3].

B. System Performance and Observations

Not only did the system maintain consistent operation, yet it also remained profitable. Despite complex conditions, the n8n workflow ran without failure - zero signal losses, outages, or timing issues occurred during testing. In nearly every run, data handling by the Python module completed in under three seconds, sometimes two, confirming speed through minimal coding solutions [4].

Despite calm stretches shrinking the range, movement patterns matched those expected from Bollinger's model on volatility [9]. With low activity, narrowing occurred - fewer alerts emerged because excessive trades were prevented. When swings intensified, broader boundaries formed through expansion, allowing room for sharp moves without false triggers. Flexibility such as this sets it apart, explaining sustained interest across research and real-world trading settings [9], [1].

Two unprofitable transactions - numbers three and six - revealed an inherent constraint in mean-reverting approaches: flawless success remains out of reach for any method. What truly matters emerges not from absolute precision, yet from whether profits on balance surpass those lost; performance here met that standard clearly through a profit factor of 6.0, demonstrating robust risk-reward dynamics even when protective mechanisms limit individual losing trades.

C. Practical Considerations

When considering ease of use, the low-code method allowed even those without coding backgrounds to engage effectively. About two hours were needed to arrange the n8n setup, most of which went into adjusting individual node settings instead of developing scripts. Such efficiency supports the idea that automated trading can become more widely available via simplified platforms.

With every entry logged into Google Sheets, traceability improved through clear records suitable for review or analysis after execution. Because each record contained timing details, categories of signals, and defined price ranges, patterns could later support refinement via algorithmic models in upcoming versions [2].

Beyond its function, n8n presents operations through a visual layout that clarifies how information moves, so tracing where issues arise becomes straightforward. Because workflows appear in a way close to natural understanding, learners grasp complex topics such as automated finance systems or artificial intelligence processes more easily within academic contexts [4].

V. DISCUSSION

A. Advantages of the Low-Code Approach

Honestly, we were exploring these low-code tools for trading has opened our eyes to the fact that it's so much more than just saving time. But, everyone start coding from scratch, but with n8n, we were able to customize it effectively without with out doing heavy kind of coding mastermind. Rather than waiting an eternity to see if something worked, we see if it worked almost instantly. But, we encountered some problems putting it out there than we normally do with regular coding.

Of course, you can't customize every little aspect, but the best part about it was how easy everything was to see. Because it's all visual, you're not stuck pouring over hundreds of lines of code. If a new person were to join our group, they could literally just look at the picture and understand what was going on. It makes the logic clear—which is a massive deal if you're trying to explain your trading logic to someone else or if you're trying to prove to a teacher (or a regulator) that your bot isn't going to go haywire [3],[4]

10.32622/ijrat.1402202604

Another really cool feature is that it's modular. It's made up of blocks, like Lego. You can test one thing, like risk management or the portfolio tools, without messing up the whole system [4] If you want to add a new feature later, you just attach it. You don't have to re-code the whole thing.

This is also a great opportunity for finance people who know what they're talking about in the market but can't code to save their souls. They can actually build and test their own models without having to beg a developer to do it for them *[3]. It just makes the whole thing faster and gets more new ideas on the table. And, n8n is compatible with pretty much any API out there, so if we wanted to switch data providers, it wouldn't be a pain. We tested it with three different market APIs and just had to change a few settings, not the whole thing [2]

B. Performance Comparison with Traditional Approaches

Development speed improves noticeably with low-code methods versus conventional programming in languages like Python or C++. Reaching deployment often demands between eight and twenty working days when building from scratch, even with skilled developers handling logic, error correction, and written guidance [3]. A different approach emerged when using the n8n platform, where setup took about two hours. This method sharply decreased coding effort by close to 95% compared to traditional development. The result stood out due to its speed, despite relying on minimal custom code.

To those without programming experience, the difference becomes especially clear. Learning enough Python to build a trading system typically takes several months. In contrast, gaining working knowledge of n8n often happens faster, using brief guides and hands-on testing [4]. Domain specialists find it easier to turn concepts into working processes quickly, since delays in sharing information tend to shrink when they build things themselves [3].

Still, watching the process unfold made problem-solving far more direct. Where mistakes appeared, the specific point of failure along with surrounding information showed clearly inside the sequence, offering insight that standard methods - dependent on paused states and layered traces - usually miss. Though silent, the interface revealed much. Still, conventional coding methods offer stronger adaptability in processing tasks along with broader scaling potential. Where advanced computations are needed - like predictive modeling, system refinement, or rapid financial transactions - these approaches allow more precision, often delivering superior speed. Despite their limitations, low-code platforms handle workflow coordination more effectively. Where processing speed or computational demand is high, conventional programming continues to offer better results.

C. Limitations and Challenges

Okay, so while the outcome was pretty sweet, we have to face reality here. We only tested this data for five days. That's fine to see if the idea is viable, but the market is a crazy place. We didn't get to see how well the bot holds up during massive market crashes, sideways markets, or massive rallies, which can really mess up reversal patterns like this [9],[1]. To be honest with ourselves, we need to backtest this thing over a few years to see how well it does in different economic cycles [12]

Also, our simulation was a bit of a "perfect world" experience. We didn't take into consideration the trade fee, slippage, or the spread. In the real world, there are trade fees. Brokers will take a cut of your trade, like 0.1% to 0.3%. That doesn't sound like much, but if you're trading like crazy, that'll cut into your profits quick. We didn't take into consideration slippage either. That's when you buy stocks for 100 rupees, but you pay 100.10 rupees because the market moved against you [12], [8]

Lastly, there's the problem of scaling. We did our tests with small amounts of fake money. In the real world, if you try to put too much money into a trade, you'll end up moving the market price against yourself. Systems that work for small amounts of money will break if you try to scale them up. So, yeah, this will work for a project, but it needs a lot more stress testing before you put real money into it [12]

VI. CONCLUSION

This study illustrates the construction of a functional algorithmic trading system utilizing Bollinger Bands and n8n workflow automation, all without the need for advanced programming expertise. The system underwent a five-day simulation on IDBI Bank (NSE), executing nine trades with a 78% success rate and generating a total simulated profit of Rs 13.3. A profit factor of around 6.0 indicates strong risk-reward dynamics, with average gains of Rs 2.3 surpassing average losses of Rs 1.5. Throughout testing, the n8n workflow experienced no failures, and Python-based signal calculations were completed in under three seconds per cycle, demonstrating the efficiency of the low-code architecture.

These results indicate that environments requiring minimal coding can effectively replicate essential algorithmic trading logic, making automated strategies accessible to those without traditional software development backgrounds. Future efforts will focus on extending backtesting periods, integrating real-time financial data feeds, and incorporating machine learning-based signal enhancements that combine multiple indicators like RSI and MACD. These advancements aim to evolve the current framework from a simulation-based proof of concept into a robust, deployable trading tool.

10.32622/ijrat.1402202604

VII. FUTURE SCOPE

A. Live Market Integration and Real-Time Execution

When a trading system goes live, it work with real market prices. Prices updates constantly, so the system has to react in fraction of seconds. There's no space for delays, and everything must stay in sync. It also needs to adjust on its own without someone stepping in to make changes. The big step happens when you move from testing in a simulated environment to trading in real financial markets. At that point, the system needs access to real-time price data and must connect directly to a brokerage platform using APIs. Trades should happen automatically, which means all parts of the system must work together smoothly and accurately.

Managing orders becomes more important in live trading. The system must handle different order types, deal with trades that only get partially filled, and keep adjusting the portfolio as market conditions change. Risk control is also critical. This includes setting clear loss limits, spreading out available funds wisely, and automatically pausing trading if the market becomes unusually volatile. Only after these protections are in place should live trading begin. That way, you can better understand whether the system performs reliably in the real world—not just in controlled testing condition

B. Machine Learning-Based Signal Enhancement

When patterns start showing a return to normal price levels, the model picks up on those signs. Instead of jumping into every possible trade, it holds back when chances look weak. Smarter predictions come through training systems to learn from past movements. By using machine learning, signal quality gets sharper over time. Sometimes it steps back to see patterns - shifts in market mood, swings in price energy, even how headlines shake things up. That way, picks come from clearer signals, not just noise. Little by little, it picks up patterns from earlier outcomes, then tweaks how it works to improve. Sometimes changes stick, sometimes they shift again based on what happens next. Smarter matters only if clarity stays. Confidence comes when people get how it works, so simplicity wins every time. What counts? Keeping things clear while adding clever touches slowly.

C. Multi-Indicator and Multi-Strategy Framework

One path involves building on Bollinger Bands through combination of further tools. Alongside them, elements like RSI, MACD, or volume shifts might sharpen entry timing while lowering misleading alerts. To avoid fighting with trends, using directional checks could keep decisions in step with overall momentum. Gradually, methods including reversion to average, movement with trends, and price breakouts might merge into one adaptable framework. Blending several styles has potential to smooth outcomes during shifting market conditions.

REFERENCES

1. Aronson, D.: Evidence-Based Technical Analysis: Applying the Scientific Method and Statistical Inference to Trading Signals. 1st Ed., John Wiley & Sons, Hoboken, 544 pages (2006). ISBN: 978-0470008744
2. Musale, J.C., Kadam, A.A.: UIM-DUDnCNN: Illumination Map-Based Denoiser and InceptionV3-Based Transfer Learning in Deep CNN for Facial Recognition. The Imaging Science Journal, Vol. 73, Issue 1, pp. 15–28 (2025). https://doi.org/10.1080/13682199.2025.2561328
3. Musale, J.C., Zad, P., Nawale, S.J., Kokare, G., Niture, S.: Stock Market Prediction Using ML. In: Multifaceted Approaches for Data Acquisition Processing and Communication, pp. 248–256. CRC Press (2024). https://doi.org/10.1201/9781003470939-32 (Book chapter)
4. Katurde, A.D., Musale, J.C., et al.: SecureSense: AI/ML Based Anomaly Detection Tool. In: Proc. IEEE Int. Conf. on Intelligent Systems for Cybersecurity (ISCS), pp. 1–6. IEEE, Gurugram (2024). https://doi.org/10.1109/ISCS61804.2024.10581060
5. Chan, E.P.: Algorithmic Trading: Winning Strategies and Their Rationale. 1st Ed., John Wiley & Sons, Hoboken, 225 pages (2013). ISBN: 978-1118460146
6. Pardo, R.: The Evaluation and Optimization of Trading Strategies. 2nd Ed., John Wiley & Sons, Hoboken, 368 pages (2008). ISBN: 978-0470128015
7. n8n.io: n8n Documentation. n8n GmbH. https://docs.n8n.io. Accessed 25 Oct 2023
8. Harris, L.: Trading and Exchanges: Market Microstructure for Practitioners. 1st Ed., Oxford University Press, New York, 643 pages (2003). ISBN: 978-0195144703
9. Bollinger, J.: Bollinger on Bollinger Bands. 1st Ed., McGraw-Hill, New York, 227 pages (2002). ISBN: 978-0071373685
10. Murphy, J.J.: Technical Analysis of the Financial Markets: A Comprehensive Guide to Trading Methods and Applications. 2nd Ed., New York Institute of Finance/Prentice Hall, New York, 576 pages (1999). ISBN: 978-0735200661
11. Lo, A.W., Hasanhodzic, J.: The Evolution of Technical Analysis: Financial Prediction from Babylonian Tablets to Bloomberg Terminals. 1st Ed., Bloomberg Press/John Wiley & Sons, New York, 224 pages (2010). ISBN: 978-1576603499
12. Lopez de Prado, M.: The 10 Reasons Most Machine Learning Funds Fail. The Journal of Portfolio Management, Vol. 44, Issue 6, pp. 120–133 (2018). https://doi.org/10.3905/jpm.2018.44.6.120

10.32622/ijrat.1402202604