

Backtesting Software Using Machine Learning

Submitted by:

Kuber Pillai, Ishan Pipara & Sahaj Singh Bhadauriya

Under the guidance of :

Mangesh Patil

I. Abstract

In today's fast-moving financial world, trading strategies need to adapt quickly to market conditions. This research focuses on building and optimizing a rule-based trading strategy using technical indicators for five well-known Indian stocks: RELIANCE, ITC, WIPRO, HDFCBANK, and TCS. By leveraging Python-based tools such as TA-Lib, yFinance, and Optuna, we created a systematic approach to trading based on the Exponential Moving Average (EMA) crossover and Relative Strength Index (RSI) conditions.

The strategy was tested on two years of historical data at an hourly interval. Initially, the strategy produced disappointing results. However, after applying Bayesian optimization through Optuna, we saw a significant turnaround in performance. Key metrics like cumulative returns, Sharpe ratio, and winning rate improved notably after optimization. The findings suggest that while traditional strategies like EMA and RSI have value, their effectiveness greatly depends on careful tuning and adaptation to the stock and timeframe being traded.

This paper presents not just the final results, but also the journey from a simple idea to a more refined and profitable system, demonstrating how even basic

strategies can be turned into powerful tools with the right approach.

II. Introduction

Financial markets are known for their complexity, unpredictability, and rapid fluctuations. Traders and investors have long relied on technical indicators to make informed decisions, with tools like Exponential Moving Averages (EMA) and the Relative Strength Index (RSI) being two of the most widely used indicators in this domain. However, while these tools provide valuable signals, their effectiveness can vary significantly depending on the asset, timeframe, and parameter values chosen.

This research was born out of a simple yet practical question: *Can we build a robust trading strategy using just a handful of popular indicators and make it profitable through optimization?* To explore this, we selected five major Indian stocks — RELIANCE, ITC, WIPRO, HDFCBANK, and TCS — representing a mix of sectors and market behaviors. The strategy was built using a logical combination of EMA and RSI, designed to generate Buy and Sell signals based on crossover and momentum conditions.

Initially, the strategy used common default values — for instance, the 14 and 21-period EMAs, and an RSI threshold of

30 and 70. These parameters, although frequently cited in trading literature, do not account for stock-specific nuances or prevailing market conditions. As a result, early backtests revealed underwhelming performance with high drawdowns and low winning rates.

To address this, we introduced a machine learning-based optimization layer using Optuna, a powerful framework for hyperparameter tuning. By treating the indicator settings as variables in a broader optimization problem, we allowed the algorithm to discover more effective combinations tailored to each stock's behavior.

The remainder of this paper details the methodology used, the logic behind the strategy design, the codebase employed, and the results obtained — both before and after optimization. Our goal is not to claim the discovery of a “holy grail” strategy, but to show how systematic tuning and backtesting can meaningfully enhance even basic rule-based trading approaches.

III. Literature Review

The use of technical analysis as a basis for trading strategies has a long history in financial markets. Over the decades, traders have leaned heavily on indicators such as the Exponential Moving Average (EMA) and the Relative Strength Index (RSI) to make sense of price trends and momentum. While these tools are widely adopted, their actual performance often hinges on how they're configured and applied.

Moving Averages, particularly EMA, have been studied extensively for their ability to

smooth out price data and react more quickly to recent price changes compared to simple moving averages (SMA). Research by Brock et al. (1992) showed that moving average crossovers could generate statistically significant returns, especially when used over longer timeframes. However, more recent studies have questioned their effectiveness in modern, high-frequency environments without proper parameter adjustments.

Similarly, RSI, introduced by J. Welles Wilder in 1978, remains one of the most popular momentum indicators. It measures the speed and change of price movements on a scale of 0 to 100. Values above 70 typically signal overbought conditions, while readings below 30 indicate oversold conditions. Several studies, such as those by Fama and French, argue that RSI by itself is not a reliable predictor of future prices, but can enhance performance when used in conjunction with other indicators or filters.

In recent years, the focus has shifted toward quantitative approaches that incorporate machine learning and optimization techniques. Tools like Optuna and Bayesian optimization frameworks allow traders and researchers to automate the process of tuning strategy parameters based on historical performance. This transition marks a shift from static rule-based systems to more data-driven adaptive models.

What this literature collectively suggests is that while technical indicators still hold value, their real-world utility is maximized when combined with intelligent optimization and backtesting frameworks. Our study positions itself at this

intersection — blending traditional tools with modern techniques to improve reliability and profitability.

IV. Methodology

This study is designed to evaluate and optimize a technical trading strategy using historical stock data from five major Indian companies: RELIANCE, ITC, WIPRO, HDFCBANK, and TCS. The methodology is divided into five main components: data collection, strategy formulation, backtesting, optimization, and performance evaluation.

I. Data Collection

We used the Yahoo Finance API (via `yfinance` in Python) to retrieve two years' worth of historical data for each selected stock. The data includes key financial variables such as Open, High, Low, Close, and Volume, sampled at a daily interval. This time frame strikes a balance between having enough observations for analysis and avoiding the noise associated with shorter time frames like intraday trading.

II. Strategy Formulation

The base trading strategy used simple yet widely respected indicators:

- Buy Condition: $EMA(close, 14) > EMA(close, 21)$ and $RSI(close, 14) > 30$
- Sell Condition: $EMA(close, 14) < EMA(close, 21)$ and $RSI(close, 14) < 70$

This setup suggests buying when the short-term EMA crosses above the long-term EMA and RSI is recovering from an oversold condition, and selling when the opposite occurs. These logic rules were parsed and executed programmatically using a custom-built strategy parser, which dynamically evaluates these expressions over the data.

III. Backtesting Engine

To test how this strategy would have performed historically, a backtesting engine was developed in Python. This engine walks through the time series data and simulates trades based on the above logic. Key metrics like entry price, exit price, trade duration, and percentage return were logged for every trade.

A split-sample approach was used:

- Training Set: 70% of data to simulate initial performance and drive optimization.
- Test Set: 30% of data to validate the optimized strategy and ensure it generalizes well.

IV. Optimization Process

Using Optuna, an open-source hyperparameter optimization framework, the strategy was subjected to Bayesian optimization across 100 trials. Each trial adjusted indicator parameters (like EMA periods and RSI lookback windows), with the goal of maximizing cumulative profit, while also considering drawdown and number of trades as constraints.

The optimization not only found better parameters but also adapted to stock-specific behaviors, effectively making the strategy more responsive and less overfit.

V. Performance Evaluation

Post-optimization, the strategy was evaluated on both training and test datasets. The following metrics were calculated:

- Cumulative Profit
- Maximum Drawdown
- Sharpe Ratio
- Winning Rate
- Average Profit per Trade
- Number of Trades

These metrics help quantify both risk and reward, offering a holistic view of the strategy's effectiveness.

V. Results and Analysis

This section presents a comparative analysis of the trading strategy's performance before and after optimization. The evaluation is based on backtested results using historical data from five Indian stocks: RELIANCE, ITC, WIPRO, HDFCBANK, and TCS, over a 2-year period.

I. Performance Before Optimization

Before applying any optimization, the trading logic was tested using default

indicator settings (EMA 14 & 21, RSI 14). The results across the training and test datasets revealed underwhelming profitability and high volatility, indicating that the strategy, in its raw form, lacked precision.

Key findings included:

- Cumulative Profit: -16.59%
- Max Drawdown: -36.50%
- Sharpe Ratio: -0.88
- Winning Rate: 32.26%
- Average Profit per Trade: -0.13%

This outcome suggested that while the trading signals were based on proven indicators, the chosen parameters were not well-tuned for the given market conditions or stock behavior.

II. Performance After Optimization

After running 100 optimization trials via Optuna, the strategy was significantly refined. The optimization focused on enhancing returns while controlling risk, which resulted in a clear improvement in all performance metrics.

Results post-optimization:

- Cumulative Profit: +8.08%
- Max Drawdown: -22.06%
- Sharpe Ratio: 0.34
- Winning Rate: 72.93%

- Average Profit per Trade: +0.06

These metrics indicate that the optimized strategy was not only more profitable but also significantly more stable. A Sharpe Ratio improvement from -0.88 to 0.34 reflects better risk-adjusted returns, while the increase in winning rate from 32.26% to 72.93% showcases enhanced consistency in trade selection.


III. Trade Volume and Consistency

Interestingly, the number of trades stayed relatively consistent (124 vs. 133), which suggests that the optimization didn't merely increase activity—it improved the quality of trade signals. This points to the efficacy of using a hybrid metric (profit, drawdown, and trade count) for guiding optimization rather than focusing solely on returns.

IV. Comparative Summary

Metric	After Optimization	Before Optimization	Improved ?
Cumulative Profit	+8.08%	-16.59%	✓ Yes
Max Drawdown	-22.06%	-36.50%	✓ Yes
Sharpe Ratio	0.34	-0.88	✓ Yes
Winning Rate	72.93%	32.26%	✓ Yes
Avg. Profit	+0.06%	-0.13%	✓ Yes

per Trade

Number of Trades 133 124  Slight Increase

The post-optimization results clearly outperform the original strategy, both in terms of profitability and stability. The drawdown control was particularly important, as it directly reflects the risk exposure a trader would face.

VI. Discussion

The findings from the backtesting and optimization experiments offer several important insights into the design, adaptability, and performance of algorithmic trading strategies based on technical indicators. This section delves into those insights and highlights the implications for both retail traders and quantitative researchers.

I. Value of Optimization in Strategy Development

One of the clearest takeaways from this research is that even a simple strategy, when left unoptimized, can lead to poor real-world outcomes. The baseline strategy—though constructed using well-known indicators like EMA and RSI—produced negative cumulative profit and high drawdowns, highlighting how crucial parameter selection is in dynamic markets.

Through optimization, especially using a tool like Optuna, these parameters were tuned to the historical behavior of selected stocks, revealing more effective

configurations that enhanced returns and reduced risk. This confirms that parameter tuning is not just a theoretical luxury—it has tangible benefits for live trading.

II. Technical Indicators Are Context Sensitive

While indicators like RSI and EMA are robust, their optimal lookback periods vary across different stocks and timeframes. For example, an EMA(14) may work well on trending stocks like RELIANCE, but may underperform on range-bound stocks like WIPRO. Without adjusting these periods to suit the context, signals may generate false positives or lag in responsiveness.

This is why the study's dynamic range approach—where sensitivity was guided by previous trial performance—proved beneficial. It helped automatically adjust indicator ranges based on their influence, leading to better-aligned signals.

III. Balancing Return and Risk

A key component of the optimization routine was its multi-objective nature: rather than optimizing only for profit, the strategy also sought to minimize drawdown and avoid overtrading. This is particularly important for retail traders who operate under capital and psychological constraints.

The study's success in improving Sharpe ratio and reducing drawdown proves that holistic optimization, rather than greed-driven returns-only metrics, can yield strategies that are both profitable and resilient.

IV. Implications for Broader Use

This framework isn't limited to the five stocks tested. Because the logic is modular and data-driven, it can be applied to:

- Other Indian equities (e.g., INFY, HINDUNILVR)
- Global markets (e.g., S&P 500, Nasdaq)
- Intraday or weekly strategies by adjusting the interval and period

Furthermore, by including more complex indicators (e.g., Bollinger Bands, Ichimoku Cloud) and introducing machine learning-based filters, future iterations can improve even further.

V. Limitations and Considerations

Despite positive results, it's important to acknowledge limitations:

- Backtesting bias: Results are based on historical data; future market conditions may differ.
- Slippage and transaction costs were not factored in. Real-world returns may be slightly lower.
- The strategy assumes perfect execution of trades on signal bars, which is rarely the case in fast-moving markets.

These challenges don't undermine the research but highlight the need for cautious interpretation and thorough paper trading or demo testing before live deployment.

VII. Conclusion and Future Work

I. Conclusion

This study presented a comprehensive approach to designing, backtesting, and optimizing technical indicator-based trading strategies using Python, TA-Lib, and Optuna. Through a modular, data-driven framework, strategies for five prominent Indian stocks—RELIANCE, ITC, WIPRO, HDFCBANK, and TCS—were evaluated over a two-year period using hourly data.

The initial results from the unoptimized strategy revealed significant limitations: high drawdown, low winning rate, and negative cumulative returns. However, once optimized using multi-objective Bayesian optimization, the same logic significantly improved in terms of profitability, risk management, and consistency. Specifically, the optimized strategy for RELIANCE alone showed an improvement from -16.59% to +8.08% in cumulative profit, with a winning rate increase from 32.26% to 72.93% and a much-improved Sharpe Ratio of 0.34.

These results reinforce the conclusion that raw strategies, no matter how logical, are rarely sufficient. Optimization, contextual calibration of indicators, and disciplined evaluation metrics are essential for robust and actionable trading systems.

II. Future Work

While the study achieved its core objective, several directions can be explored to enhance and expand the research:

- **Include Transaction Costs:** Adding slippage, brokerage fees, and other trading costs will help assess strategy viability in real-world scenarios.
- **Incorporate Stop-Loss and Take-Profit Rules:** These can help manage risk more explicitly and may improve performance during volatile markets.
- **Explore Ensemble Strategies:** Combining multiple strategies using voting or weighting mechanisms can reduce overfitting and improve robustness.
- **Machine Learning Filters:** Incorporating ML models like XGBoost or LSTM to act as filters before executing technical indicator signals could further refine entries.
- **Cross-Asset Application:** Applying the framework to other asset classes such as commodities, currencies, or crypto can validate its adaptability.
- **Real-Time Deployment:** Integrating the system with live feeds and brokers via APIs (like Zerodha Kite or Alpaca) could enable live paper trading or deployment.

In summary, this work lays the foundation for algorithmic trading education and experimentation, especially for Indian retail investors and aspiring quantitative analysts. It demonstrates that with the right

tools, even simple strategies can be elevated into powerful systems—given the discipline to test, optimize, and iterate.

VIII. References

1. Talib Documentation – *Technical Analysis Library in Python (TA-Lib)*. Retrieved from: <https://mrjbjq7.github.io/ta-lib/>
2. Optuna: A Hyperparameter Optimization Framework – Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). *Optuna: A Next-generation Hyperparameter Optimization Framework*. Proceedings of the 25th ACM SIGKDD. doi:10.1145/3292500.3330701
3. Yahoo Finance API via yfinance – Open-source library for market data. GitHub Repository: <https://github.com/ranaroussi/yfinance>
4. Pandas Documentation – Python Data Analysis Library. <https://pandas.pydata.org/>
5. NumPy Documentation – Scientific computing tools for Python. <https://numpy.org/>
6. Joblib Library – Efficient serialization of Python objects. <https://joblib.readthedocs.io/>
7. Backtrader Framework (Optional Reference) – While not used directly, offers relevant comparisons for custom backtesting systems. <https://www.backtrader.com/>
8. Kaufman, P. J. (2013). *Trading Systems and Methods*. John Wiley & Sons.
9. Murphy, J. J. (1999). *Technical Analysis of the Financial Markets: A Comprehensive Guide to Trading Methods and Applications*. New York Institute of Finance.
10. Python Software Foundation. Python Language Reference, version 3. <https://www.python.org/>