International Journal of Scientific Engineering and Research (IJSER)
ISSN (Online): 2347-3878
SJIF (2025): 8.036

When the Market Changes Its Mind: Regime Fragility in Machine-Learning Forecasts of a Simulated NSE Index

Evaan Behal

The Millennium School, Amritsar, Punjab, India
Email: *evaan0007[at]gmail.com*

Abstract: ***Machine-learning models are widely promoted as tools for predicting stock prices, and many published studies report directional accuracies well above chance for Indian indices such as the NIFTY 50. Most of these studies test one model on one historical sample, which makes it hard to tell whether a reported edge reflects genuine skill or a favourable historical period. This paper takes a different route. I built a synthetic market calibrated to the broad statistical behaviour of a large-cap National Stock Exchange (NSE) index, in which the true amount of predictability is known and can be switched up or down. The market alternates between a calm regime, where short-term trends tend to continue, and a turbulent regime, where they tend to reverse. Four models (logistic regression, random forest, gradient boosting and a neural network) were trained with a walk-forward design on 120 independent simulated paths, each containing approximately eleven trading years and then on a further 600 paths spanning six levels of signal strength. All four models captured only 20 to 41 percent of the accuracy that was theoretically available, and every one of them was right less than half the time on turbulent days. Adding explicit regime information did not help. The simplest model, logistic regression, performed best, yet after realistic trading costs it beat a passive buy-and-hold strategy in fewer than one path in five. The results suggest that regime instability, rather than model sophistication, is the binding constraint on AI-driven market prediction.***

Keywords: stock market prediction, machine learning, NSE, NIFTY 50, regime switching, Monte Carlo simulation, backtesting

1. Introduction

Few questions in finance attract as much attention or generate considerable speculation as whether computers can predict the stock market. Over the past decade the question has moved from academic journals into everyday life. Retail participation on the National Stock Exchange of India has grown sharply, trading apps advertise algorithmic signals, and social media is full of screenshots showing neural networks that apparently forecast the NIFTY 50 with remarkable precision. At the same time, a long line of financial theory argues that easy, persistent predictability should not exist in a liquid market, because traders who spot it would trade it away (Fama, 1970; Malkiel, 2003).

The research literature sits somewhere between these two positions. Several well-cited studies report that machine-learning models can forecast the direction of index movements with accuracy above 50 percent, including studies of Indian indices (Patel et al., 2015). Larger studies using carefully controlled methods have found real but modest predictive power, concentrated in particular periods and market conditions (Fischer & Krauss, 2018; Gu et al., 2020). The trouble is that almost all of this evidence comes from a single historical sample. When a model looks good on 2010 to 2020 data, we cannot rerun history to see whether the result would hold in a different decade, and we never know how much predictability was actually there to be found.

This paper addresses that problem by using simulation. Instead of asking whether a model can predict the real NIFTY 50, it asks a more controlled question: if a market behaves broadly like an NSE index and contains a small, known amount of predictability that changes with market conditions, how much of that predictability can standard machine-learning models actually recover, and how much survives trading costs? Because the data are synthetic, the experiment can be repeated hundreds of times, the true answer is known in advance, and the influence of each ingredient can be isolated. The focus on changing market conditions, or regimes, is deliberate. Markets are widely believed to behave differently in calm and stressed periods (Ang & Bekaert, 2002; Hamilton, 1989), yet most forecasting studies treat the data as if it came from a single stable process.

2. Literature Review

2.1 Market Efficiency and Its Critics

The efficient market hypothesis holds that prices already reflect available information, so past prices should not reliably predict future returns (Fama, 1970). In its strict form the hypothesis leaves no room for technical forecasting at all. Later work softened this view. Malkiel (2003) acknowledged that short-lived patterns appear but argued that they are usually too small or too unstable to exploit after costs. Timmermann and Granger (2004) made a related point from the forecasting side: any pattern that becomes widely known should decay, so successful forecasting models are likely to be fragile and short-lived. Lo (2004) proposed the adaptive markets hypothesis, in which the degree of efficiency itself varies over time as market participants, conditions and competition change. This idea is central to the present study, because it implies that predictability may exist in some periods and vanish, or even reverse, in others.

2.2 Statistical Properties of Returns

Any realistic simulation needs to reproduce the main statistical features of real returns. Mandelbrot (1963) showed that price changes have far heavier tails than a normal distribution would predict. Cont (2001) summarised a set of

Licensed Under Creative Commons Attribution CC BY

stylised facts that hold across many markets: daily returns are close to uncorrelated, but their magnitudes are strongly autocorrelated, so large moves cluster together; distributions are fat-tailed; and volatility tends to rise when prices fall. The GARCH model introduced by Bollerslev (1986) captures volatility clustering by letting today's variance depend on yesterday's shock and yesterday's variance. Hamilton (1989) introduced Markov regime switching, in which an unobserved state variable moves between distinct regimes with their own parameters. Combining the two approaches gives a flexible model of markets that alternate between quiet and stressed periods.

2.3 Machine Learning for Market Prediction

Machine-learning methods have been applied to return forecasting for decades, but interest accelerated after improvements in tree ensembles and neural networks. Random forests (Breiman, 2001) and gradient-boosted trees (Chen & Guestrin, 2016) can capture non-linear interactions between features without the analyst specifying them. Long short-term memory networks (Hochreiter & Schmidhuber, 1997) were designed for sequential data and have been applied widely to price series. Krauss et al. (2017) compared deep networks, gradient-boosted trees and random forests on S&P 500 constituents and found economically meaningful returns before costs, with a sharp decline in later years. Fischer and Krauss (2018) found that an LSTM outperformed simpler models but that its advantage faded after about 2010, which they attributed partly to markets becoming more efficient. In the most comprehensive study to date, Gu et al. (2020) found that machine-learning methods improved return prediction relative to linear benchmarks, with most of the gain coming from non-linear interactions. For India, Patel et al. (2015) reported high directional accuracy for the CNX Nifty and S&P BSE Sensex using technical indicators. A systematic review by Sezer et al. (2020) noted that much of the deep-learning literature uses short samples, inconsistent evaluation and little attention to trading costs.

2.4 Backtest Overfitting

A growing body of work warns that impressive backtests are often illusions. Bailey et al. (2014) showed mathematically that when researchers try many model configurations on one dataset, the best-looking result is likely to be overfitted and to disappoint out of sample. López de Prado (2018) argued that standard machine-learning practices, such as random cross-validation, leak future information into training when applied to financial time series. Formal tests for comparing forecasts, such as those of Diebold and Mariano (1995) and Pesaran and Timmermann (1992), help, but they still rely on a single realised history.

2.5 Research Gap

Taken together, the literature suggests that predictability in equity indices is small, unstable and easily overstated. However, two things are rarely measured directly. First, studies on real data cannot say what fraction of the available predictability a model captured, because the true amount is unknown. Second, although regime changes are widely discussed, few studies test what happens when the direction of a pattern reverses between regimes, rather than merely weakening. A controlled simulation calibrated to an Indian index can address both points.

3. Research Question and Hypotheses

The central research question is: *In a market calibrated to the broad behaviour of an NSE large-cap index, where short-term predictability exists but changes sign between calm and turbulent regimes, how much of the available predictive edge can common machine-learning models capture, and does that edge survive realistic trading costs?*

Four hypotheses follow from the literature. H1 states that models will capture well under half of the theoretically available accuracy. H2 states that accuracy will be significantly lower on turbulent days than on calm days, because the dominant calm-regime pattern is learned and then misapplied when the regime changes. H3 states that giving models explicit information about the current regime will reduce this gap. H4 states that more flexible models (random forest, gradient boosting and the neural network) will not outperform logistic regression when the signal is weak, consistent with the overfitting concerns raised by Bailey et al. (2014).

4. Methodology

4.1 Synthetic NSE-Calibrated Data

Each simulated path contains 2,750 trading days, roughly eleven years at 250 trading days per year. On each day the market is in one of two hidden regimes. In the calm regime the probability of staying calm the next day is 0.985; in the turbulent regime the probability of staying turbulent is 0.95. These values produce calm spells lasting a few months on average and turbulent spells lasting a few weeks, broadly in line with the episodic character of Indian market stress periods. Daily variance follows a GARCH(1,1) process with parameters 0.08 and 0.90, pulled toward a regime-specific long-run level equal to a daily volatility of 0.85 percent in calm periods and 1.9 percent in turbulent ones. Shocks are drawn from a Student-*t* distribution with five degrees of freedom, scaled to unit variance, to generate fat tails.

The predictable component is the key design choice. Let $m_{5,t}$ denote the mean return over the previous five days. The conditional expected return is defined as:

$$\mu_t = \mu_{st} + \varphi_{st} \times \sqrt{h_t} \times \text{sign}(m_{5,t})$$

where μ_t is the conditional expected return at time t, μ_{st} is the drift associated with the current regime, h_t is the conditional variance, and φ_{st} is the regime-dependent predictable-component coefficient. In the calm regime, $\varphi_{st} = \beta$, so recent trends tend to continue. In the turbulent regime, $\varphi_{st} = -1.5\beta$, so recent trends tend to reverse. If the preceding five-day mean is zero, the predictable component is zero.

The regime-dependent drift is slightly positive in the calm regime and slightly negative in the turbulent regime. The baseline signal strength is $\beta = 0.05$, which is deliberately small. Specifically, the regime-dependent drift is $\mu_0 = 0.0007$ in the calm regime and $\mu_1 = -0.0008$ in the turbulent regime.

Licensed Under Creative Commons Attribution CC BY

Daily conditional variance follows a regime-dependent GARCH(1,1) process:

$$h_t = \omega_{st} + \alpha \times r_{t-1}^2 + \gamma \times h_{t-1}$$

where h_t is the conditional variance, r_{t-1} is the preceding day's return, $\alpha = 0.08$ is the coefficient on the preceding squared return, $\gamma = 0.90$ is the coefficient on the preceding conditional variance, and ω_{st} is the regime-dependent variance intercept.

The variance intercept is defined as:

$$\omega_{st} = (1 - \alpha - \gamma) \times \sigma_{st}^2$$

where σ_{st} is the target daily volatility for the current regime: 0.0085 in the calm regime and 0.019 in the turbulent regime.

The realised return is generated as:

$$r_t = \mu_t + \sqrt{h_t} \times \sqrt{((\nu - 2)/\nu)} \times z_t$$

where z_t follows a Student-t distribution with $\nu = 5$ degrees of freedom. The scaling factor ensures that the innovation has unit variance before multiplication by the conditional standard deviation.

The simulation begins in the calm regime, with an initial return of zero and an initial conditional variance of $h_0 = 0.0085^2$. The regime evolves according to a two-state Markov chain, with probabilities of remaining in the calm and turbulent regimes of 0.985 and 0.95, respectively.

Table 1 compares the resulting statistics with the general properties expected of a large-cap index.

Table 1: Statistical Properties of the Simulated Index (Averages Over 120 Paths)

Property	Simulated value	Expected behaviour of a large-cap index
Annualised return	10.80%	Positive, roughly 8–15%
Annualised volatility	18.70%	Roughly 15–22%
Excess kurtosis (median)	7.1	Clearly positive (fat tails)
Lag-1 autocorrelation of returns	0.006	Close to zero
Lag-1 autocorrelation of squared returns	0.14	Clearly positive (volatility clustering)
Share of test days in turbulent regime	23%	A minority of days
Average maximum drawdown	−41.7%	Large drawdowns occur in most decades

Note. Ranges in the right-hand column are approximate and are intended to describe the general shape of large-cap equity index behaviour rather than precise NIFTY 50 statistics.

Figure 1 shows a simulated index path over 2,750 trading days, with turbulent regimes shaded and the beginning of the out-of-sample testing period marked by a dashed vertical line.

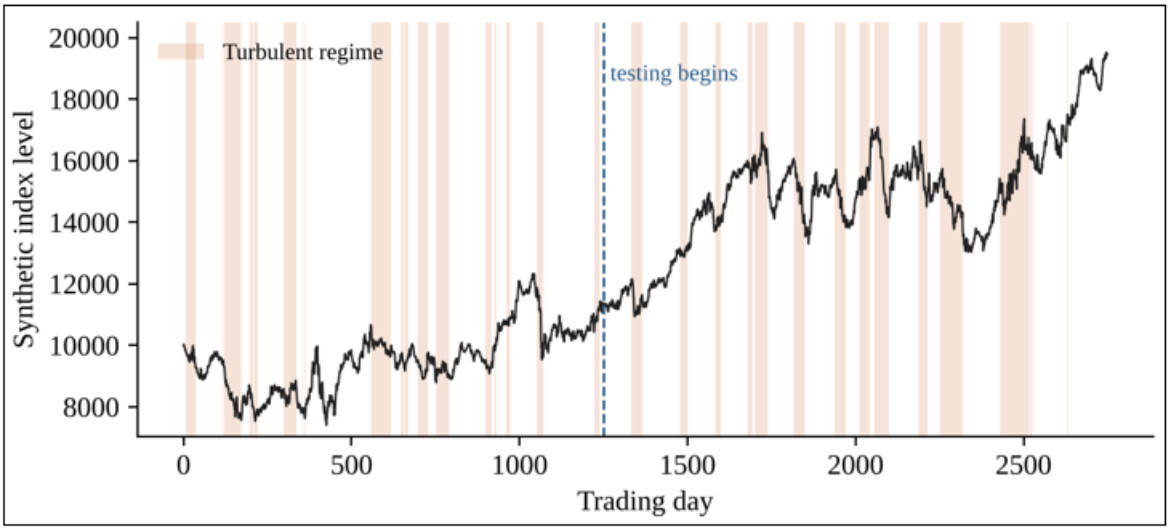


Figure 1: One Simulated Path (2,750 Trading Days) With Turbulent Regimes Shaded.

4.2 Features

Models were given nine features that require no information beyond the price series, in the style of the technical indicators used by Patel et al. (2015): the three most recent daily returns, five-day and twenty-day average returns, twenty-day realised volatility, the ratio of five-day to sixty-day volatility, the distance of the index from its fifty-day moving average, and the sign of the five-day average return. The target was whether the next day's return was positive. For the regime-aware variant, two further features were added: a high-volatility flag equal to one when twenty-day volatility exceeded its median up to that date, and the product of this flag with the five-day sign. Only past data were used for every feature, so no future information could leak into the forecasts.

4.3 Models

Four models were compared, all implemented in scikit-learn (Pedregosa et al., 2011): logistic regression with standardised inputs; a random forest of 150 trees with maximum depth five; a histogram-based gradient-boosting classifier with 150 shallow trees; and a feed-forward neural network with two hidden layers of 32 and 16 units, weight decay and early stopping. Hyperparameters were fixed in advance and not tuned on test data. Recurrent networks such as LSTMs were not included because their computational cost across hundreds of paths was impractical for this project, and because the simpler network already allows the question of model complexity to be tested.

Licensed Under Creative Commons Attribution CC BY

4.4 Walk-Forward Testing and Evaluation

Each simulated path contains 2,750 observations. The initial training boundary is set at index 1,250, corresponding approximately to five trading years at 250 observations per year. Model fitting begins at index 60 because the feature set requires historical observations for rolling-window calculations.

The first training sample therefore uses indices 60–1,249, and the first out-of-sample test window uses indices 1,250–1,499.

The model is subsequently retrained at the beginning of each 250-observation test window, using an expanding training sample.

The successive test windows are indices 1,250–1,499, 1,500–1,749, 1,750–1,999, 2,000–2,249, 2,250–2,499, and 2,500–2,748.

The final observation at index 2,749 is excluded from evaluation because its subsequent realised return is unavailable.

This procedure produces 1,499 out-of-sample predictions per path. The first 60 observations are excluded from model fitting to accommodate the historical windows required for feature construction; they are not additional test observations. The resulting test period covers approximately six trading years, following the initial training period.

Benchmark Calculations and Trading Costs

The oracle benchmark represents an idealised forecasting strategy that knows the sign of the simulated conditional expected return on each test day. It takes a long position when the conditional expected return is positive and a short position when it is negative. Because this benchmark relies on information unavailable to the predictive models, it serves as a theoretical reference for the predictability present in the simulation rather than as a directly implementable trading strategy.

Directional accuracy is calculated as the proportion of test observations for which the predicted direction agrees with the realised direction. The oracle's accuracy is calculated using the sign of the simulated conditional expected return and the corresponding realised return. The capture ratio measures the proportion of the oracle's directional-accuracy advantage over chance that is recovered by a predictive model. It is defined as: Capture Ratio = (Model Accuracy − 0.50) / (Oracle Accuracy − 0.50) An accuracy of 0.50 represents chance-level directional prediction. The capture ratio is meaningful when the oracle's accuracy exceeds 0.50; otherwise, the denominator is zero or negative, requiring separate interpretation.

For trading evaluation, each model takes a position of +1 when its predicted probability of a positive return exceeds 0.50 and −1 otherwise. Daily gross strategy returns are calculated by multiplying the position by the subsequent realised return.

Transaction costs are deducted according to the absolute change in position:

$$\text{Net Return}_t = \text{Position}_t \times \text{Return}_{t+1} - c \times |\text{Position}_t - \text{Position}_{t-1}|$$

where $c = 0.0005$ represents the assumed transaction cost per unit of turnover. The initial position is zero. Consequently, opening a long or short position incurs a cost of 5 basis points, while switching directly from long to short incurs a cost of 10 basis points.

The annualised Sharpe ratio is calculated from daily strategy returns as the mean daily return divided by the standard deviation of daily returns, multiplied by the square root of 250. Gross Sharpe ratios use returns before transaction costs, whereas net Sharpe ratios use returns after transaction costs.

The buy-and-hold benchmark remains continuously invested in the index. Its returns are calculated separately from the active trading strategies and are not reduced by the transaction-cost adjustment applied to the model-generated positions.

4.5 Experiments

Experiment 1 ran all four models, in both regime-blind and regime-aware form, on 120 independent paths at the baseline signal strength. Experiment 2 varied β across six levels from 0 to 0.10, using 100 new paths per level (600 in total), and compared logistic regression and gradient boosting. Differences between variants were tested with paired *t*-tests across paths.

5. Results

5.1 Overall Accuracy and Profitability

Table 2 summarises Experiment 1. The oracle achieved 54.2 percent accuracy and a gross Sharpe ratio of 1.19, which shows that the baseline market did contain a meaningful edge. None of the models came close to it. Logistic regression was the most accurate at 51.8 percent, followed by the random forest at 51.6 percent, the neural network at 51.1 percent and gradient boosting at 50.9 percent. Expressed as capture ratios, the models recovered between 20 and 41 percent of the available edge, which supports H1.

Table 2: Out-of-Sample Results Across 120 Paths at Baseline Signal Strength (β = 0.05, Regime-Blind Models)

Strategy	Accuracy	Capture ratio	Gross Sharpe	Net Sharpe	Net Sharpe > 0	Beats buy-and-hold
Oracle	54.20%	100%	1.19	—	—	—
Buy-and-hold	52.50%	—	0.62	0.62	—	—
Logistic regression	51.80%	41%	0.4	0.12	62%	19%
Random forest	51.60%	37%	0.32	−0.03	45%	11%
Neural network	51.10%	27%	0.27	−0.10	41%	12%
Gradient boosting	50.90%	20%	0.16	−0.35	27%	8%

Licensed Under Creative Commons Attribution CC BY

Note. Accuracy and Sharpe ratios are averages across paths. The last two columns show the percentage of paths in which the model's net Sharpe ratio was positive, and in which it exceeded buy-and-hold's.

A striking detail is that buy-and-hold was more accurate than every model, at 52.5 percent, simply because the index rose on slightly more than half of all days. An accuracy above 50 percent therefore says very little on its own; it can be achieved by employing no active forecasting strategy. After costs the picture became worse. Only logistic regression had a positive average net Sharpe ratio (0.12), and it was positive in 62 percent of paths. Gradient boosting traded the most, changing position on roughly a third of all days, and its costs turned a small gross profit into a clear loss. No model beat buy-and-hold in more than about one path in five.

5.2 Performance Across Regimes

Figure 2 separates accuracy by regime. On calm days every model was right more often than not, with logistic regression reaching 52.9 percent. On turbulent days every model was below 50 percent, ranging from 47.7 percent for logistic regression to 49.3 percent for gradient boosting. Pooling all models and paths, the gap between calm and turbulent accuracy was highly significant, $t(959) = 30.9$, $p < .001$, which supports H2.

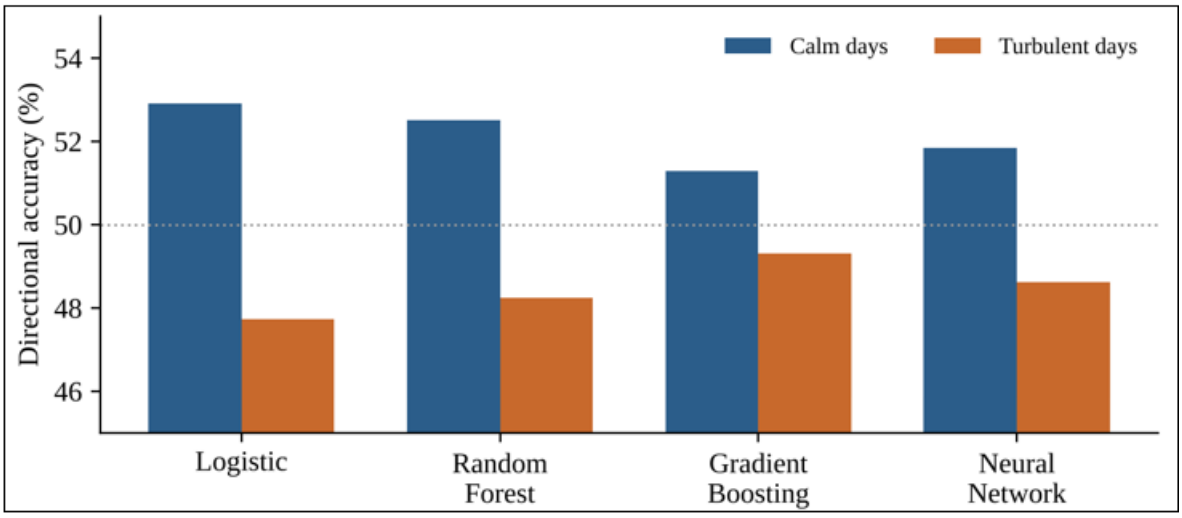


Figure 2: Directional Accuracy on Calm and Turbulent Days (Average of 120 Paths, Regime-Blind Models)

The explanation is visible in the design. Calm days make up about three-quarters of the data, so the dominant pattern the models learn is momentum. When the market turns turbulent, the same pattern points the wrong way, and the models are confidently wrong exactly when the stakes are highest. Interestingly, the model that learned momentum best, logistic regression, also suffered the most on turbulent days, while gradient boosting, which learned less of anything, lost less. Doing well in one regime appears to come at the direct cost of doing badly in the other.

5.3 Does Regime Information Help?

H3 was not supported. Adding the high-volatility flag and its interaction with recent trend made almost no difference. For the random forest, gradient boosting and neural network, the changes in accuracy, turbulent-day accuracy and net Sharpe ratio were all small and statistically insignificant (all $p > .27$). For logistic regression, regime information raised turbulent-day accuracy by about half a percentage point, $t(119) = 2.65$, $p = .009$, but slightly lowered overall accuracy, $t(119) = -2.20$, $p = .030$, and net Sharpe ratio, $t(119) = -2.24$, $p = .027$. In other words, the volatility flag was too noisy a guide to the hidden regime. Regimes are identified only after volatility has already risen, by which time some of the turbulent spell has passed, and calm days with briefly high volatility are misclassified.

5.4 How Much Signal Is Needed?

Figure 3 shows the results of Experiment 2. As β increased from 0 to 0.10, the oracle's gross Sharpe ratio rose from 1.02 to 1.75, while logistic regression's net Sharpe ratio rose only from −0.07 to 0.22. Its net performance was significantly above zero only when β reached 0.08 ($p = .002$), with $\beta = 0.06$ marginal ($p = .053$). Gradient boosting was loss-making after costs at every signal level, with average net Sharpe ratios between −0.37 and −0.25. Even at the strongest signal tested, logistic regression beat buy-and-hold in only 19 percent of paths. This pattern supports H4: extra model flexibility did not help when the signal was weak, and it cost more in trading.

Licensed Under Creative Commons Attribution CC BY

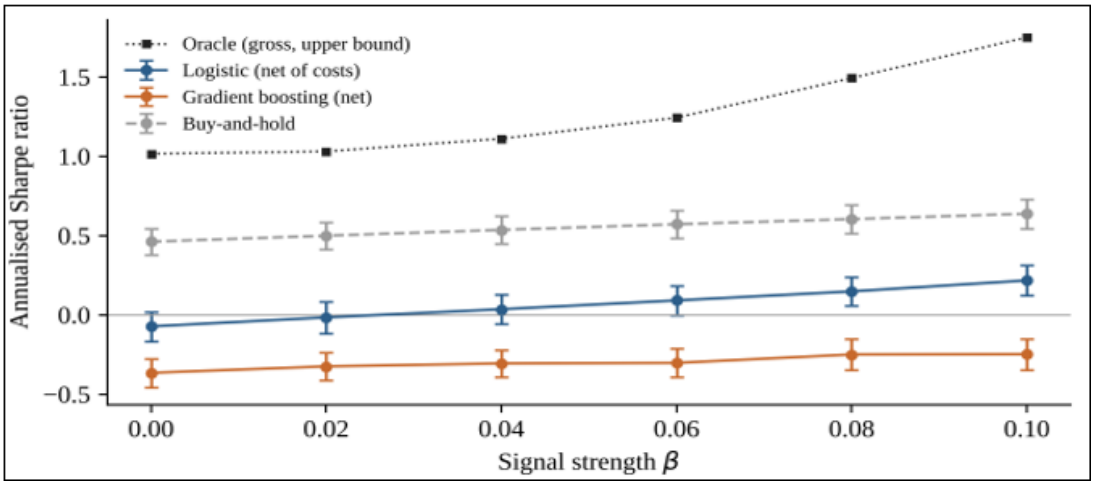


Figure 3: Net Sharpe Ratio as Signal Strength Increases (100 Paths per Level; Error Bars Show 95% Confidence Intervals)

A useful reading of the zero-signal case (β = 0) is that the oracle still achieved 53.9 percent accuracy, because the two regimes have different average drifts. The models picked up part of this drift and reached about 51 percent accuracy without any trend pattern to exploit, yet produced no profit after costs. This is a clear warning against judging a forecasting model by its hit rate alone.

6. Discussion

The main finding is that regime instability, not lack of model power, limits what these models can achieve. The simulated market contained a real and fairly generous edge, and an oracle could have earned a Sharpe ratio nearly twice that of buy-and-hold. Yet the models captured less than half of it, because the edge was not one pattern but two opposite patterns depending on conditions the models could not see directly. This links to Lo's (2004) adaptive markets hypothesis: if predictability shifts with conditions, a model trained on an average of those conditions will be tuned to the most common one and performs poorly under less frequently observed regimes.

The results also help interpret the published evidence. Studies reporting high accuracy for Indian indices, such as Patel et al. (2015), used a single historical sample. In this simulation the spread of results across paths was large; the standard deviation of net Sharpe ratios across paths was between 0.45 and 0.51 for every model, so some individual paths produced excellent backtests purely by chance. A researcher who happened to observe one of those paths would reach a far more optimistic conclusion than the average result supports. This is exactly the problem described by Bailey et al. (2014), and it argues for reporting results across many samples or resamples rather than a single backtest.

The weak performance of the more complex models is consistent with Fischer and Krauss (2018) and Krauss et al. (2017), who found that advantages for sophisticated models shrank over time and after costs. It may seem to conflict with Gu et al. (2020), who found clear gains from non-linear methods. However, their setting was the cross-section of thousands of stocks, where many weak signals can be combined; here the task was timing a single index, where there is far less information to learn from. The design of this study also deliberately made the signal simple, a single sign-flipping rule, so a linear model was well suited to it. In a market with richer interactions the ranking might change.

For an investor on the NSE, the practical implication is modest but useful. A model that claims to predict the NIFTY 50 should be judged against buy-and-hold, not against 50 percent; its accuracy should be checked separately in calm and volatile periods; and its trading costs should be included before any profits are claimed. In this study, a model passing all three checks was rare even when a real edge was known to exist.

7. Limitations

The most important limitation is that the data are simulated. The process was calibrated to reproduce general features of a large-cap index, but it was not fitted to actual NIFTY 50 prices, and real markets contain features not modelled here, such as overnight gaps, policy announcements, foreign investor flows and changes in market structure. The results therefore describe how models behave in an NSE-like environment, not how they would have performed on the real index. A natural next step would be to fit a Markov-switching GARCH model to historical NIFTY 50 data and repeat the experiments using those estimated parameters. Second, the predictable component was designed by the researcher, and other patterns might favour different models. Third, models were not tuned and recurrent networks were not tested, so a better-tuned or sequence-based model might perform somewhat better. Finally, the trading rule was simple and the cost assumption approximate; position sizing based on forecast confidence could reduce turnover and costs.

8. Conclusion

This study asked how much predictive edge machine-learning models can recover from an NSE-style market in which predictability exists but reverses between calm and turbulent regimes. Across 720 simulated paths, the answer was: not much. Models captured between a fifth and two-fifths of the available accuracy, were systematically wrong during turbulent periods, did not benefit from a simple regime indicator, and rarely beat buy-and-hold after costs. The simplest model performed best. These findings do not show that AI cannot help in financial markets, but they do suggest

Licensed Under Creative Commons Attribution CC BY

that the critical challenge is recognising when the rules of the market have changed, and that claims of reliable AI-driven index prediction deserve careful scrutiny. Future work could test whether better regime detection, such as a hidden Markov model estimated in real time, can close part of the gap between the models and the oracle.

References

[1] Ang, A., & Bekaert, G. (2002). International asset allocation with regime shifts. *The Review of Financial Studies, 15*(4), 1137–1187. https://doi.org/10.1093/rfs/15.4.1137

[2] Bailey, D. H., Borwein, J. M., López de Prado, M., & Zhu, Q. J. (2014). Pseudo-mathematics and financial charlatanism: The effects of backtest overfitting on out-of-sample performance. *Notices of the American Mathematical Society, 61*(5), 458–471. https://doi.org/10.1090/noti1105

[3] Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics, 31*(3), 307–327. https://doi.org/10.1016/0304-4076(86)90063-1

[4] Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

[5] Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785–794). ACM. https://doi.org/10.1145/2939672.2939785

[6] Cont, R. (2001). Empirical properties of asset returns: Stylized facts and statistical issues. *Quantitative Finance, 1*(2), 223–236. https://doi.org/10.1080/713665670

[7] Diebold, F. X., & Mariano, R. S. (1995). Comparing predictive accuracy. *Journal of Business & Economic Statistics, 13*(3), 253–263. https://doi.org/10.1080/07350015.1995.10524599

[8] Fama, E. F. (1970). Efficient capital markets: A review of theory and empirical work. *The Journal of Finance, 25*(2), 383–417. https://doi.org/10.2307/2325486

[9] Fischer, T., & Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research, 270*(2), 654–669. https://doi.org/10.1016/j.ejor.2017.11.054

[10] Gu, S., Kelly, B., & Xiu, D. (2020). Empirical asset pricing via machine learning. *The Review of Financial Studies, 33*(5), 2223–2273. https://doi.org/10.1093/rfs/hhaa009

[11] Hamilton, J. D. (1989). A new approach to the economic analysis of nonstationary time series and the business cycle. *Econometrica, 57*(2), 357–384. https://doi.org/10.2307/1912559

[12] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[13] Krauss, C., Do, X. A., & Huck, N. (2017). Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500. *European Journal of Operational Research, 259*(2), 689–702. https://doi.org/10.1016/j.ejor.2016.10.031

[14] Lo, A. W. (2004). The adaptive markets hypothesis: Market efficiency from an evolutionary perspective. *The Journal of Portfolio Management, 30*(5), 15–29. https://doi.org/10.3905/jpm.2004.442611

[15] López de Prado, M. (2018). *Advances in financial machine learning*. Wiley.

[16] Malkiel, B. G. (2003). The efficient market hypothesis and its critics. *Journal of Economic Perspectives, 17*(1), 59–82. https://doi.org/10.1257/089533003321164958

[17] Mandelbrot, B. (1963). The variation of certain speculative prices. *The Journal of Business, 36*(4), 394–419. https://doi.org/10.1086/294632

[18] Patel, J., Shah, S., Thakkar, P., & Kotecha, K. (2015). Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques. *Expert Systems with Applications, 42*(1), 259–268. https://doi.org/10.1016/j.eswa.2014.07.040

[19] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12*, 2825–2830.

[20] Pesaran, M. H., & Timmermann, A. (1992). A simple nonparametric test of predictive performance. *Journal of Business & Economic Statistics, 10*(4), 461–465. https://doi.org/10.1080/07350015.1992.10509922

[21] Sezer, O. B., Gudelek, M. U., & Ozbayoglu, A. M. (2020). Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Applied Soft Computing, 90*, Article 106181. https://doi.org/10.1016/j.asoc.2020.106181

[22] Timmermann, A., & Granger, C. W. J. (2004). Efficient market hypothesis and forecasting. *International Journal of Forecasting, 20*(1), 15–27. https://doi.org/10.1016/S0169-2070(03)00012-8

Licensed Under Creative Commons Attribution CC BY

Appendix: Reproducibility

The simulation and forecasting experiments were implemented in Python using NumPy, pandas, and scikit-learn. The complete simulation script is provided below, including the regime-switching data generator, feature construction, model definitions, expanding-window evaluation procedure, and performance calculations.

For the baseline experiment, random seeds ranged from 0 to 119, corresponding to 120 simulated paths. The signal-strength experiment evaluated six values of β: 0, 0.02, 0.04, 0.06, 0.08, and 0.10, with seeds 1000–1099 used for each signal-strength level.

The simulation parameters, including regime-transition probabilities, regime-dependent drifts, target variances, GARCH coefficients, model settings, and transaction costs, are specified in the supplementary script. Each simulated path is generated using NumPy's random-number generator, initialised with the corresponding seed.

The software environment used to test the script comprised Python 3.14.0, NumPy 2.5.4, pandas 3.1.0, scikit-learn 1.9.1, and SciPy 1.18.1.

Simulation Code

```
"""
Regime fragility in ML forecasts of an NSE-calibrated index.
Monte Carlo study: synthetic NIFTY-50-like returns with Markov regime switching,
GARCH(1,1) volatility, Student-t shocks and a weak regime-dependent predictable
component (momentum in calm regime, reversal in turbulent regime).
"""
import numpy as np, pandas as pd, sys, time, json
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, HistGradientBoostingClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
import warnings; warnings.filterwarnings("ignore")

N_DAYS = 2750        # ~11 trading years (250 days/yr)
TRAIN0 = 1250        # initial training window (5 yrs)
STEP = 250           # retrain yearly
COST = 0.0005        # 5 bps per unit of turnover (one side)

# regime parameters (0 = calm, 1 = turbulent), daily units
P_STAY = np.array([0.985, 0.95])
MU = np.array([0.0007, -0.0008])
OMEGA = np.array([0.8e-6, 4.0e-6])
ALPHA, BETA_G = 0.08, 0.90
TARGET_VAR = np.array([0.0085**2, 0.019**2])
T_DF = 5

def simulate(beta, rng, n=N_DAYS):
    s = np.zeros(n, int); r = np.zeros(n); h = np.zeros(n); pred = np.zeros(n)
    s[0] = 0; h[0] = TARGET_VAR[0]
    tscale = np.sqrt((T_DF - 2) / T_DF)
    for t in range(1, n):
        s[t] = s[t-1] if rng.random() < P_STAY[s[t-1]] else 1 - s[t-1]
        # GARCH variance, pulled toward regime target
        om = TARGET_VAR[s[t]] * (1 - ALPHA - BETA_G)
        h[t] = om + ALPHA * r[t-1]**2 + BETA_G * h[t-1]
        sig = np.sqrt(h[t])
        m5 = r[max(0, t-5):t].mean() if t > 1 else 0.0
        phi = beta if s[t] == 0 else -1.5 * beta
        # predictable part scaled to volatility so signal strength is comparable
        pred[t] = MU[s[t]] + phi * sig * np.sign(m5) if m5 != 0 else MU[s[t]]
        r[t] = pred[t] + sig * tscale * rng.standard_t(T_DF)
    return r, s, pred

def features(r):
    x = pd.Series(r)
    f = pd.DataFrame({
        "lag1": x, "lag2": x.shift(1), "lag3": x.shift(2),
```


Licensed Under Creative Commons Attribution CC BY

```
        "mom5": x.rolling(5).mean(), "mom20": x.rolling(20).mean(),
        "vol20": x.rolling(20).std(), "volratio": x.rolling(5).std() / x.rolling(60).std(),
        "ma50gap": (1 + x).cumprod() / (1 + x).cumprod().rolling(50).mean() - 1,
        "sign5": np.sign(x.rolling(5).mean()),
    })
    # regime proxy: 20-day vol above its expanding median (no look-ahead)
    f["hivol"] = (f["vol20"] > f["vol20"].expanding(60).median()).astype(float)
    f["sign5_x_hivol"] = f["sign5"] * f["hivol"]
    return f

BASE = ["lag1","lag2","lag3","mom5","mom20","vol20","volratio","ma50gap","sign5"]
AWARE = BASE + ["hivol","sign5_x_hivol"]

def model(name, seed):
    if name == "Logistic": return make_pipeline(StandardScaler(), LogisticRegression(C=1.0, max_iter=500))
    if name == "RandomForest": return RandomForestClassifier(n_estimators=150, max_depth=5, min_samples_leaf=20,
n_jobs=1, random_state=seed)
    if name == "GradBoost": return HistGradientBoostingClassifier(max_depth=3, learning_rate=0.05, max_iter=150,
random_state=seed)
    if name == "NeuralNet": return make_pipeline(StandardScaler(), MLPClassifier((32,16), alpha=1e-3, max_iter=300,
early_stopping=True, random_state=seed))

def walk_forward(f, y, cols, name, seed):
    probs = np.full(len(y), np.nan)
    for start in range(TRAIN0, N_DAYS - 1, STEP):
        tr = slice(60, start); te = slice(start, min(start + STEP, N_DAYS - 1))
        m = model(name, seed); m.fit(f[cols].values[tr], y[tr])
        probs[te] = m.predict_proba(f[cols].values[te])[:, 1]
    return probs

def evaluate(probs, fut, s_next, pred_next):
    idx = ~np.isnan(probs)
    pos = np.where(probs[idx] > 0.5, 1.0, -1.0)
    ret = fut[idx]
    gross = pos * ret
    turn = np.abs(np.diff(np.concatenate([[0], pos])))
    net = gross - COST * turn
    acc = np.mean(pos == np.sign(ret))
    ann = lambda z: np.sqrt(250) * z.mean() / z.std()
    calm = s_next[idx] == 0
    return dict(acc=acc, acc_calm=np.mean((pos == np.sign(ret))[calm]),
                acc_turb=np.mean((pos == np.sign(ret))[~calm]),
                sharpe_gross=ann(gross), sharpe_net=ann(net), turnover=turn.mean())

def run_path(beta, seed, names):
    rng = np.random.default_rng(seed)
    r, s, pred = simulate(beta, rng)
    f = features(r)
    fut = np.append(r[1:], np.nan); s_next = np.append(s[1:], 0); p_next = np.append(pred[1:], 0)
    y = (fut > 0).astype(int)
    out = []
    idx_test = np.arange(TRAIN0, N_DAYS - 1)
    # oracle: knows true expected return sign
    orc = np.sign(p_next[idx_test]); orc[orc == 0] = 1
    out.append(dict(seed=seed, beta=beta, model="Oracle", variant="-",
                    acc=np.mean(orc == np.sign(fut[idx_test])),
                    sharpe_net=np.nan, sharpe_gross=np.sqrt(250)*(orc*fut[idx_test]).mean()/(orc*fut[idx_test]).std()))
    bh = fut[idx_test]
    out.append(dict(seed=seed, beta=beta, model="BuyHold", variant="-", acc=np.mean(bh > 0),
                    sharpe_gross=np.sqrt(250)*bh.mean()/bh.std(), sharpe_net=np.sqrt(250)*bh.mean()/bh.std()))
    for name in names:
        for var, cols in (("blind", BASE), ("aware", AWARE)):
            p = walk_forward(f, y, cols, name, seed)
```


Licensed Under Creative Commons Attribution CC BY

```
            d = evaluate(p, fut, s_next, p_next); d.update(seed=seed, beta=beta, model=name, variant=var)
            out.append(d)
    return out, r, s

if __name__ == "__main__":
    mode = sys.argv[1]
    t0 = time.time(); rows = []
    if mode == "main":
        for seed in range(int(sys.argv[2]), int(sys.argv[3])):
            o, _, _ = run_path(0.05, seed, ["Logistic","RandomForest","GradBoost","NeuralNet"]); rows += o
        pd.DataFrame(rows).to_csv(f"main_{sys.argv[2]}.csv", index=False)
    elif mode == "sweep":
        for beta in [0.0, 0.02, 0.04, 0.06, 0.08, 0.10]:
            for seed in range(1000, 1000 + int(sys.argv[2])):
                o, _, _ = run_path(beta, seed, ["Logistic","GradBoost"]); rows += o
        pd.DataFrame(rows).to_csv("sweep.csv", index=False)
    elif mode == "time":
        o, _, _ = run_path(0.05, 0, ["Logistic","RandomForest","GradBoost","NeuralNet"])
        print(pd.DataFrame(o)[["model","variant","acc","acc_calm","acc_turb","sharpe_gross","sharpe_net"]].round(3))
    print("secs", round(time.time() - t0, 1))
```


Licensed Under Creative Commons Attribution CC BY