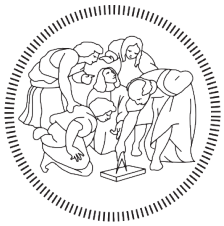


POLITECNICO
MILANO 1863

Nifty50 Forecasting

June 15, 2025

Barbieri Claudio
Delera Giacomo
Hauser Cyril
Howe Alessandro

Contents

1 Introduction and Literature Review

The Indian stock market, recently characterized by rapid growth, has garnered significant attention from investors. Among its various indices, the Nifty 50 stands out as a benchmark, representing the performance of 50 major companies listed on the National Stock Exchange of India. This index serves as a barometer for the Indian equity market and is widely used for investment purposes.

As of 2025, both institutional investors - including hedge funds and asset management firms - and retail investors have shown substantial interest in the Nifty 50 index. This also thanks to the popularity of Nifty 50 Exchange-Traded Funds, that offer low-cost access to a diversified portfolio of India's leading companies.

The aim of this study is to investigate whether it is possible to develop effective forecasting models for the NIFTY 50 index. To this end, we explore a range of time-series modeling techniques, starting with traditional linear regression and progressing to more advanced approaches such as Conv-LSTM, VAR and GARCH models. Additionally, in some occasions we incorporate also information of the U.S. market to see if the results improve. Our ultimate goal is to leverage these model signals to develop a trading strategy on the index.

There is a substantial body of literature on financial forecasting. First, classical econometric models—such as linear regressions, VAR and related multivariate techniques and GARCH—are often employed to forecast OHLC and return variables (e.g. [4]). Second, a growing number of studies apply machine learning and deep neural networks, including LSTM, CNN and hybrid architectures (e.g., [1]). In this work, we investigate both approaches to blend the traditional econometric models with the learning power of neural networks. See bibliography for full references.

2 Data

We have high-frequency (minute-level) data from the Indian stock market, covering the period from January 2, 2017, to January 1, 2021. The data is sourced from the National Stock Exchange of India (NSE) and includes a comprehensive set of time series along with a master file containing metadata such as trading symbol, instrument type, segment, and a unique identifier for each file.

These are publicly available on Kaggle and can be accessed here[1].

The data is organized into three main categories:

- **Index Data:** This includes minute-level data for the NIFTY 50, NIFTY 100, and NIFTY 500 indices. Each record contains a timestamp, open, high, low, close prices, and volume. Notably, the volume for index data is recorded as zero, as indices themselves are not directly traded instruments.
- **Sectoral Indices:** There are six sector-specific indices included: NIFTY AUTO, NIFTY BANK, NIFTY COMMODITIES, NIFTY FIN SERVICE, NIFTY ENERGY and NIFTY MIDCAP 100 with the same structure as the general indices. Also here, volume is recorded as zero.
- **Constituent Stocks:** The dataset includes minute-level data for the individual stocks that comprise the NIFTY 50, 50 additional stocks from NIFTY 100, and 50 more from NIFTY 150. Unlike indices, these records include actual traded volumes alongside price data. Each stock file includes the same five core variables: timestamp, open, high, low, close, and volume.

We also use the SPDR S&P 500 ETF Trust (SPY) as a practical proxy for the S&P 500 index. This can be requested through the website API ALPHA VANTAGE[2]. The file contains daily values of open, high, low, and close prices of the fund along with the traded volume.

2.1 Processing

Minute data usually shows a high level of noise that can affect the reliability of models fitted to it. Moreover, it is more likely to have possible missing values for some minutes due to technical problem or other issues in the collecting phase (as in 2017-07-10 when a major glitch crippled the NSE). Finally, most of the articles we can find in the literature for our purposes work on daily data. These facts lead us to aggregate our observations. We take all the data from a day and ensure to consider only the ones

[1] https://www.kaggle.com/code/debashis74017/trading-strategy-using-5-min-nifty-50-index-data?select=NIFTY+50+-+Minute+data.csv

[2] https://www.alphavantage.co/

corresponding to times in which the Indian market is open, that is, from 9:15 to 15:30 IST. The open price is the one detected at 9:15, and the close price is the one of 15:30. Then, the high and low prices are selected by looking at the maximum and minimum value among all the minute values. Since we are analyzing an index that is not traded on the market, we cannot observe the transaction volume. To have a proxy that can be used in the computation of some technical indicators, we extract it from the underlying stock components. With Nifty50 volume, we refer to the sum of the daily number of trades of the 50 largest Indian companies listed on the National Stock Exchange, which are its constituents.

A further preliminary selection that we make is to focus on data from 2017, 2018, and 2019. This choice is due to the advent of the COVID-19 pandemic in 2020, which significantly changed the market behavior in a way that no model can reasonably predict, and so the goodness of it would be biased.

3 Methodology and Results

3.1 Linear Regression with Technical Indicators

We initially tackle the prediction of the daily closing price of the Nifty50 index, addressing it as a supervised learning problem. As a starting point, we employ linear regression, a widely used baseline model in financial literature due to its simplicity, interpretability and effectiveness in capturing linear trends in time series data. Moreover, linear regression serves as a useful benchmark before exploring more complex models. Here our focus is purely on predictive performance, aiming to gain a general overview of the forecasting capabilities. We are aware that we have a non-stationary target variable and we sideline the check of the classical linear model assumption needed to asses the statistical significance of the results. We emulate the logic of complex machine learning methods in which several of these hypotheses do not need to be verified, and see how it affects the performance when translated to simple linear regression and have a general overview of the predictive performance.

Most of these models consider technical indicators as possible features able to explain how the price will move in the future. This idea is also confirmed by the fact that commonly trading strategies leverage them. Among the wide range of available technical indicators, we selected a subset of those proposed in [2].

Category	**Indicators**
Trend-following	MACD (Moving Average Convergence/Divergence), DEMA (Double Exponential Moving Average), CP (Close Price)
Momentum (Oscillators)	ROC (Rate of Change), RSI (Relative Strength Index), ULTOSC (Ultimate Oscillator), WR (Williams %R), CCI (Commodity Channel Index)
Volatility	ATR (Average True Range), VOL (Volatility)
Volume-based	OBV (On Balance Volume)

Table 1: Classification of technical indicators used in this study.

To build a proper dataframe on which to fit a model, avoiding any temporal leakage and look-ahead bias, we computed the indicators on the lagged data. This means that the features we extract to forecast future values y_{t+h} come only from data available at time t. To compute the indicators, we use the TTR package in R, which already provides some functions to get them given an OHLCV time series. Given this setting, our trial is not only to predict the next-day closing price but also to assess the model's forecasting accuracy for horizons of 3, 5 and 7 days ahead. Since the underlying patterns and dependencies in financial time series can vary across different forecast horizons, we allow ourselves to fit distinct models for each horizon (1-, 3-, 5-, and 7-day ahead). This flexibility acknowledges that the features and dynamics most relevant for short-term prediction may not be equally informative for longer-term forecasts, and vice versa.

A visualization that can be helpful to understand the framework is the following:

$$y_{t+1} = \mathbf{X}_{t,1}\beta_1 \qquad y_{t+3} = \mathbf{X}_{t,3}\beta_3 \qquad y_{t+5} = \mathbf{X}_{t,5}\beta_5 \qquad y_{t+7} = \mathbf{X}_{t,7}\beta_7$$

where y_{t+i} is the close price i days ahead, $\mathbf{X}_{t,i}$ is the matrix of technical indicators selected for time horizon i using data up to time t and β_i the corresponding coefficient.

Being in a supervised setting, we have to split the data and use part of it for the parameter estimation, and the other part to evaluate the model's performance on unseen data. The split strategy follows the standard practice in time series modeling, where past observations are used to train the model and future

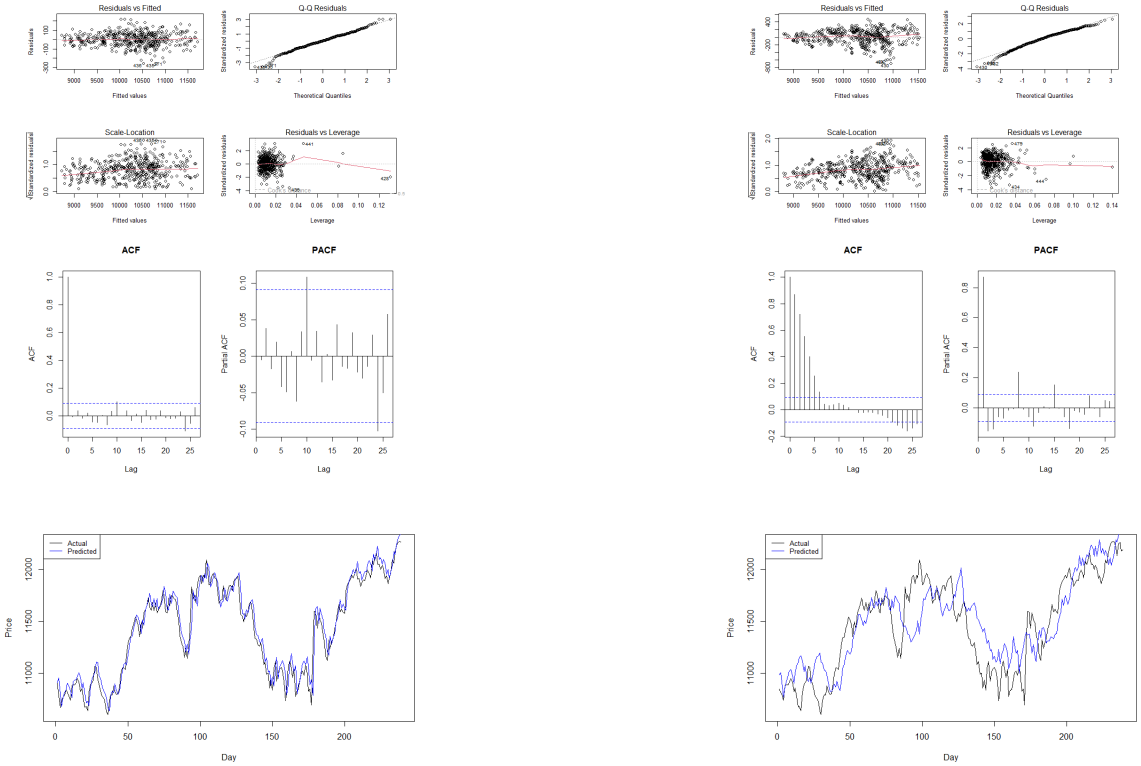
observations are used for testing, preserving the temporal order of the data. In our specific case, the 2017 and 2018 data are used for training, and the 2019 data for testing.

The feature selection for each model fit is made following the standard procedure of forward and backward selection method (as in [3]) and than choose the one minimizing the AIC between the two.

The whole procedure gives us these final models:

$$\begin{aligned}
\text{CLOSE}_{t+1} &= \alpha_1 + \text{CLOSE}_t + \text{ULTISC}_t + \text{ATR}_t + \text{OBV}_t + \text{WR}_t \\
\text{CLOSE}_{t+3} &= \alpha_3 + \text{CLOSE}_t + \text{OBV}_t + \text{VOL}_t + \text{ULTISC}_t + \text{WR}_t + \text{MACD}_t \\
\text{CLOSE}_{t+5} &= \alpha_5 + \text{CLOSE}_t + \text{MACD}_t + \text{VOL}_t + \text{ROC}_t + \text{DEMA}_t + \text{OBV}_t + \text{WR}_t + \text{ULTISC}_t \\
\text{CLOSE}_{t+7} &= \alpha_7 + \text{MACD}_t + \text{ULTISC}_t + \text{VOL}_t + \text{DEMA}_t + \text{OBV}_t + \text{WR}_t + \text{ATR}_t
\end{aligned}$$

where with α_i we indicate the presence of an intercept.



(a) Model 1 step ahead

(b) Model 7 step ahead

Figure 1: Diagnostic plots for Models 1 and 7 step-ahead predictions.

Day Ahead	Adj R-squared	DW test	BP test	JB test	RMSE
1	0.9881	p-value $= 0.4736$	p-value < 0.01	p-value < 0.01	106.6839
3	0.9631	p-value < 0.01	p-value < 0.01	p-value < 0.01	196.3811
5	0.9374	p-value < 0.01	p-value < 0.01	p-value < 0.01	234.7391
7	0.9163	p-value < 0.01	p-value < 0.01	p-value < 0.01	260.0908

We can see how the prediction accuracy decreases as the prediction horizon grows, and so does the correlation of the residuals. The shifted behavior of the prediction with respect to the actual values can be a symptom of a leading component of the current price in the forecast.

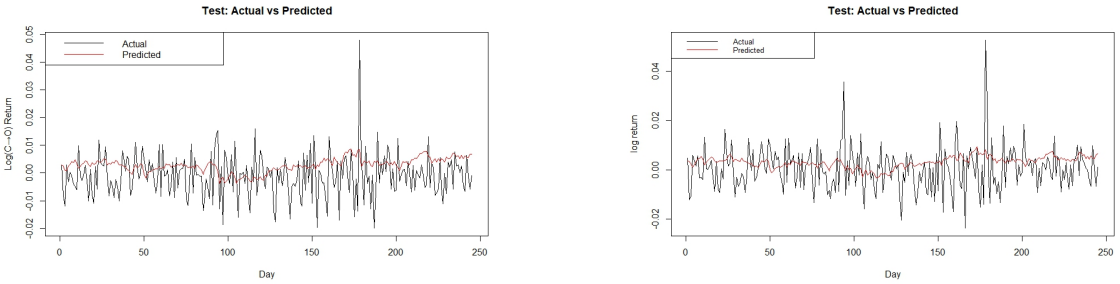
Generally, as expected, the residual diagnostics highlight clear violations of CLRM assumptions: the significant PACF lags beyond lag0 and the Durbin–Watson test reject no-autocorrelation; the Breusch–Pagan test and residual plots signal heteroscedasticity, indicating volatility clustering; the Jarque–Bera test, alongside heavy-tailed QQ plots, confirms non-normality.

We want to stress again that these results are not intended for scenarios requiring statistical significance or inferential reliability. Our initial approach does not aim to provide statistically rigorous

conclusions. Rather, we use these models as an exploratory tool to gain preliminary insights into the predictive potential of various technical indicators and to identify directions for building more robust models. We are aware of the classical limitations of applying linear regression to time series data, including autocorrelated residuals, inflated R^2 values, and the risk of spurious regression, and thus treat the current results as heuristic and suggestive rather than definitive.

...on Log Returns

Aware of the usual setting used in econometrics, we attempt to apply the same methodology to predict the log return. For our purposes, we model both $\log(\text{close}_t/\text{open}_t)$ and $\log(\text{close}_{t+1}/\text{close}_t)$. Unlike the raw close price time series, both of these return series are stationary (the ADF test returns a p-value $<$ 0.01 in both cases). Moreover, the fit of this model exhibits a more theoretical validity since the CLMR assumptions are met. The plots below display the model's predictions over the 2019 period for both settings.



(a) $\log(\text{close}_t/\text{open}_t)$ (b) $\log(\text{close}_{t+1}/\text{close}_t)$

Figure 2: Forecast results of the linear model.

3.2 Trading Strategies

To assess how effective our models are at generating profitable signals, we first implement three simple trading strategies (see LSTM Python notebook). In each case, we assume a transaction cost of 5bps whenever we enter or exit a position in the index.

We apply these strategies under two distinct assumptions. In the first scenario, we assume no positions (long or short) are held overnight during market-closed hours. Thus, if a position is maintained across consecutive trading days, transaction costs are incurred both at the prior day's close and the following day's open. In this setup the model's prediction target is the intraday return, $\log(\text{close}_t/\text{open}_t)$. In the second scenario, we assume positions are held continuously, including overnight. This means the strategy is exposed to market movements during closed hours—i.e., the difference between the previous day's close and the following day's open. In this case, the model predicts the standard log-return: $\log(\text{close}_{t+1}/\text{close}_t)$.

With a slight abuse of notation, we will use the term *log return* to refer to both $\log(\text{close}_t/\text{open}_t)$ and $\log(\text{close}_{t+1}/\text{close}_t)$, depending on the context. The three implemented strategies are as follows:

1. **Absolute Change Threshold Strategy.**
   We first compute the day-to-day relative change in the model's predicted log-returns:
   $$\Delta_t = \left| \frac{\hat{r}_t - \hat{r}_{t-1}}{\hat{r}_{t-1}} \right|.$$
   Whenever Δ_t exceeds a fixed threshold (here, 10%), we take a new position: long if $\hat{r}_t > 0$, short if $\hat{r}_t < 0$, and flat if $\hat{r}_t = 0$. If Δ_t is below the threshold, we simply maintain our prior position.

2. **Fixed Threshold Sign Strategy.**
   In this simpler variant, we compare each predicted log-return $\hat{r}_t$ against a small positive and negative threshold (e.g. $\pm 0.01\%$). We go long whenever $\hat{r}_t >$ threshold, short whenever $\hat{r}_t <$ $-$threshold, and remain flat otherwise.

3. **Volatility-Scaled Strategy.**
   Here, we adjust our exposure based on recent market volatility. We compute a rolling standard

deviation of the true log-returns over a lookback window (e.g. 10days), shifted by one day to avoid look-ahead bias. The raw model signal $\hat{r}_t$ is then scaled by the inverse of this volatility:

$$\text{position}_t = \text{clip}\Big(\kappa \, \frac{\hat{r}_t}{\sigma_{t-1} + \varepsilon} \, , \, -L, \, +L \Big),$$

where κ is a user-defined scale factor, L is the maximum leverage and ε prevents division by zero.

Ultimately, we choose to adopt the Fixed Threshold Sign Strategy for model evaluation as it seems to be the best responsive to directional signals.

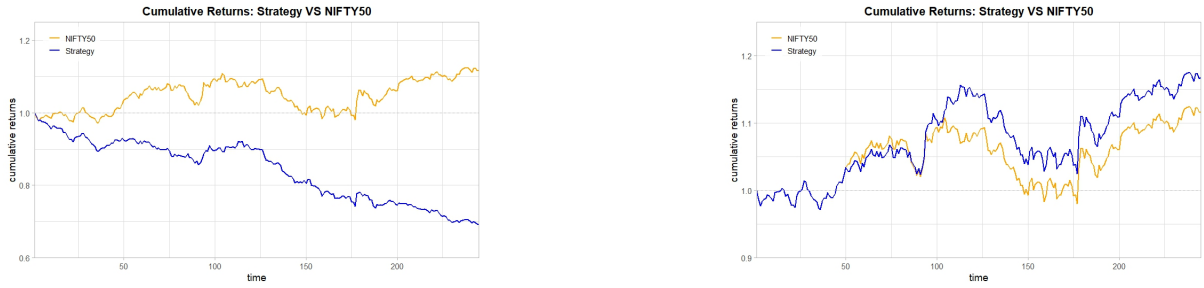
Both of the underlying assumptions, however, come with their own limitations. In the intra-day setting—where the model predicts $\log(\text{close}_t/\text{open}_t)$—the predictive performance is generally better. However transaction costs accumulate rapidly (we generally pay these twice a day), and the strategy is eaten by these costs. Conversely, in the overnight case—where the model forecasts $\log(\text{close}_{t+1}/\text{close}_t)$—transaction costs are lower, but the model tends to perform worse in capturing the true log return dynamics.

3.2.1 Results for the linear model

We apply the Fixed Threshold Sign Strategy to our linear regression model (on 2019), once predicting the intraday log-return $\log(\text{close}_t/\text{open}_t)$ and once predicting the one-step-ahead log-return $\log(\text{close}_{t+1}/\text{close}_t)$. The performance metrics are summarized in Table2.

Target	Ann. Return	Sharpe	Max DD	Sortino
$\log(\frac{\text{C}}{\text{O}})$	-31.34%	-3.00	30.79%	-4.98
$\log(\frac{\text{C}_{t+1}}{\text{C}_t})$	17.79%	1.24	11.38%	2.40

Table 2: Performance metrics for the linear model under the Fixed Threshold Strategy.



(a) Intraday trading.

(b) Whole day trading.

Figure 3: Cumulative strategy vs. NIFTY50 for the linear model.

When targeting the intraday return, the strategy is "eaten" by transaction costs. In contrast, forecasting the standard one-step log-return $\log(\text{close}_{t+1}/\text{close}_t)$ produces positive results, outperforming the index itself.

3.3 LSTM with Technical Indicators

Given the wide use of LSTM networks in the literature for financial forecasting (e.g. [1]), we implement a Convolution-LSTM (Conv-LSTM) architecture to predict the log-returns. In this setting, we focus only on a one-step ahead prediction. After experimenting with several variants, we adopt the following structure:

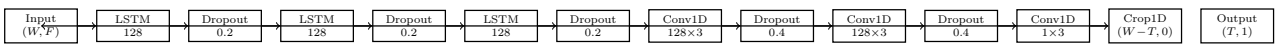


Figure 4: Conv-LSTM architecture

First of all we convert our historical series into a sliding-window tensor:

$$\mathbf{X} \in \mathbb{R}^{N \times W \times F}, \quad \mathbf{y} \in \mathbb{R}^{N \times 1},$$

where each sample consists of W consecutive days of F features (the previous day's log-return plus its technical indicators), and the target is the next day's log-return. Two hyperparameters control this encoding:

- *Window size* W: the number of past days observed for each prediction.
- *Stride*: the step length by which the window advances along time (here set to 1 for daily forecasts).

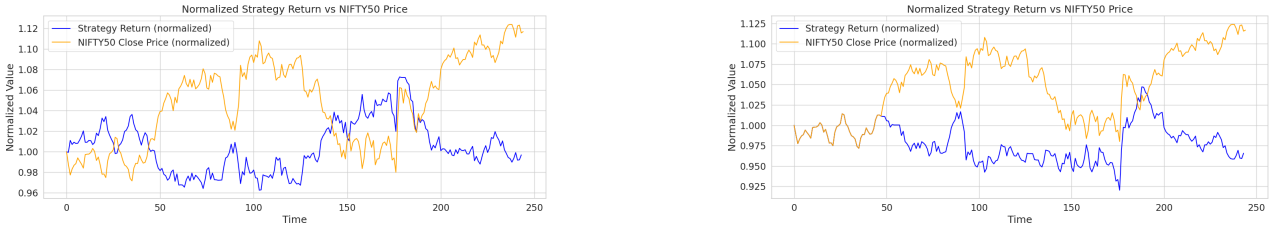
Although we implement a "telescope" parameter for many-to-many prediction, in this study we fix it to 1 for a many-to-one (day-ahead) output.

We build each sequence for the training set (from the start of the series up to 30 Sep 2018), the validation set (1 Oct 2018–31 Dec 2018) and the test set (all 2019). To ensure each window is complete, we extend the validation and test periods backwards by W days. Before training, we apply Min–Max scaling (fit on the training split) to all features and targets.

After applying the Fixed Threshold Sign Strategy to our Conv–LSTM model, we obtain the performance metrics summarized in Table3.

Target	**Ann. Return**	**Sharpe**	**Max DD**	**Sortino**
$\log(\frac{\text{close}_t}{\text{open}_t})$	-0.06%	$+0.06$	7.90%	$+0.09$
$\log(\frac{\text{close}_{t+1}}{\text{close}_t})$	-3.12%	-0.16	9.49%	-0.28

Table 3: Performance metrics for the Conv–LSTM model under the Fixed Threshold Strategy.



(a) Intraday trading

(b) Whole day trading

Figure 5: Cumulative strategy vs. NIFTY50 for the Conv–LSTM model.

The Conv–LSTM model yields nearly flat performance when predicting intraday returns, with a small positive Sharpe but very low signal. It can be also noticed that forecasting overnight returns deteriorates performance.

3.3.1 Addition of S&P 500 Data

We further investigate whether incorporating U.S. market information improves our forecasts. The trading hours of the two markets do not overlap:

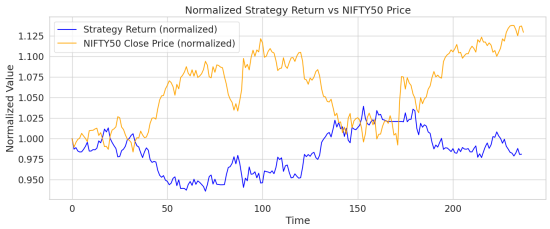
- **Nifty 50 (NSE, India)** operates Monday–Friday, opening at 9:15AM IST and closing at 3:30PM IST.
- **S&P 500 (NYSE, US)** trades Monday through Friday, opening at 9:30 am ET and closing at 4:00 pm ET. In IST, this corresponds to 7:00PM–1:30AM IST (during EDT), or 8:00PM–2:30AM IST (during EST).

Since NYSE trading ends several hours before the NSE opens, we can safely use SPY's prior-day log returns and trading volume as additional features. We retrain the same Conv-LSTM architecture considering also the new features. The results are shown in Table4.

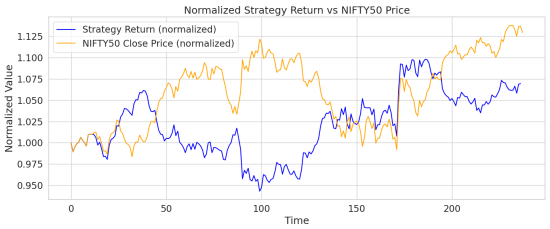
Adding SPY prior-day returns and volume improves overnight predictions but intraday performance remains negative. The addition of the U.S. market signals contributes modestly to the performance, but it does not lead to a significant improvement in overall strategy performance. We therefore do not take it into account from now on.

Target	Ann. Return	Sharpe	Max DD	Sortino
$\log(\frac{\text{close}_t}{\text{open}_t})$	-2.91%	-0.21	7.60%	-0.34
$\log(\frac{\text{close}_{t+1}}{\text{close}_t})$	$+5.95\%$	$+0.49$	11.13%	$+0.78$

Table 4: Performance metrics for the Conv–LSTM model with SPY features.



(a) Intraday trading with SPY features.



(b) Whole day trading with SPY features.

Figure 6: Cumulative strategy vs. NIFTY50 after adding SPY data.

3.4 VAR on the OHLC

Following the idea that technical indicators can capture market dynamics, and considering that they are computed from the multivariate time series of **O**pen, **H**igh, **L**ow and **C**lose prices (and volume, but not included here), we want to directly model them. We think that relaxing their contribution from a fixed formula to a learnable combination of their values can help the prediction ability.

The method we use is a Vector AutoRegressive (VAR) that allows us to jointly model the time series. The procedure we implement is inspired by the work of [4], which introduces the data transformation needed to ensure the constraints for OHLC prices are satisfied. Since we are dealing with prices, all the values must be positive. Moreover, open and close prices must be within the range of the low and high, which must be a valid range.

Thus, if we denote as $x_t^o, x_t^h, x_t^l, x_t^c$ the variable of the original time series, the transformed version we deal with comes from the following formulas[2]:

$$y_t^{(1)} = \log x_t^l \quad y_t^{(2)} = \log(x_t^h - x_t^l) \quad y_t^{(3)} = \log \frac{\lambda_t^o}{1-\lambda_t^o} \quad y_t^{(4)} = \log \frac{\lambda_t^c}{1-\lambda_t^c}$$

With:

$$\lambda_t^c = \frac{x_t^c - x_t^l}{x_t^h - x_t^l} \quad \lambda_t^o = \frac{x_t^o - x_t^l}{x_t^h - x_t^l}$$

Note that these expressions work under the assumption that prices at a given t are all different between the series; otherwise, it would lead to indefinite values. In the real financial market, this is not always the case, but we face it with some minor random adjustments.

The multivariate time series modeled by a VAR is

$$\mathbf{Y}_t = (y_t^{(1)}, y_t^{(2)}, y_t^{(3)}, y_t^{(4)})^T$$

from which we can retrieve the prices, which are what we are interested in, by inversion of the above formulas.

In the described framework, we decide to test how well this model can perform in forecasting. We set a sliding window $\mathbf{Y}_1^{(q)}$ of q days used to train the VAR, and then use it to predict the next m days. An illustration for the procedure for a single window is in 7.

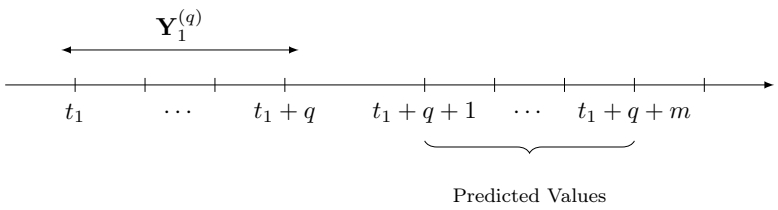


Figure 7: General windowed forecast prediction

[0](for the detailed explaination of these choices please refere to the orginal work)

Now let's focus on the fitting procedure for a single window, that is, a model of the form

$$\mathbf{Y}_t = \mathbf{A}_1\mathbf{Y}_{t-1} + \ldots + \mathbf{A}_p\mathbf{Y}_{t-p} + \mathbf{e}$$

where $\mathbf{e}$ is the error term. The lag p is adaptively chosen according to the multivariate information criterion; in this case, we used the MAIC. The validity of the VAR depends on the stationarity of all the univariate time series components, which is why we check it any time with the ADF test, and in case the assumption is not verified, we differentiate the corresponding one and keep track of this. In this way, we can build the prediction starting from the last observed value and combine it with the predicted differences.

$$y_{t+m} = y_t + \sum_{i=1}^{m} \Delta y_{t+i}$$

We scroll the window one step forward and repeat the operation. In the end, we gather all the results and compute the performances using RMSE so we can compare them to the ones of the simple linear model. We test the model for different window lengths, and from the diagnostics of correlation, heteroskedasticity and normality, we opt for $q = 90$. This allows for no autocorrelation in the residuals and few cases of non-heteroskedasticity, at the cost that the normality assumption is rarely met. However, since our primary objective is to generate point forecasts rather than conduct formal statistical inference, the violation of the normality assumption does not compromise the validity of our results. In particular, VAR forecast estimates are consistent and reliable under standard conditions even when residuals deviate from normality.

Here we report the results for different time horizons $m = 1, 3, 5, 7$

	$m = 1$	$m = 3$	$m = 5$	$m = 7$
$x^{(o)}$	60.7086	111.2485	146.7744	175.0846
$x^{(h)}$	72.2246	115.6153	147.9248	174.8228
$x^{(l)}$	76.5855	128.2882	162.6039	188.9592
$x^{(c)}$	95.5690	134.4361	164.4519	189.3141

Table 5: RMSE results for $q = 90$ and $m = 1, 3, 5, 7$

We observe a clear improvement over the previous models, particularly when evaluating forecast performance over an extended horizon. This suggests that the revised specification better captures the underlying dynamics of the system, enabling more accurate and stable forecasts beyond the short term.

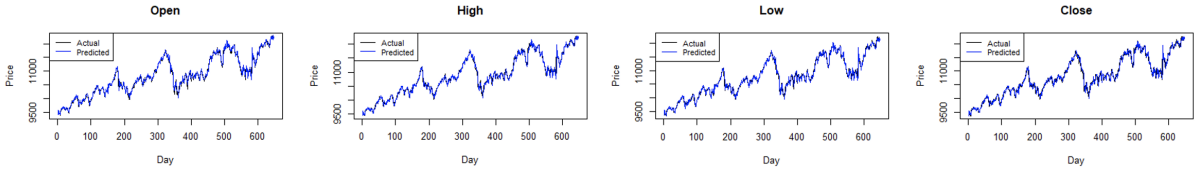


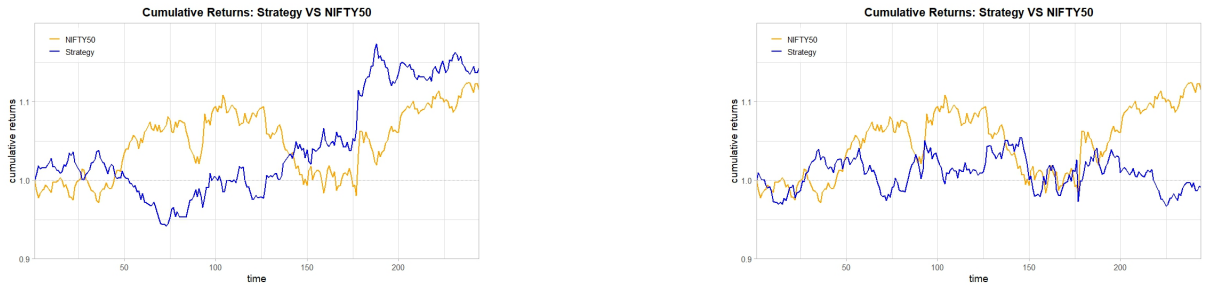
Figure 8: One-step VAR forecasts on transformed OHLC variables (m=1).

The trading strategy results, based on the VAR model with $m = 1$ and using both $\log(\text{close}_t/\text{open}_t)$ and $\log(\text{close}_{t+1}/\text{close}_t)$ computed from the forecasts, are reported below.

Target	**Ann. Return**	**Sharpe**	**Max DD**	**Sortino**
$\log(\frac{\text{close}_t}{\text{open}_t})$	14.35%	1.17	9.33%	2.28
$\log(\frac{\text{close}_{t+1}}{\text{close}_t})$	−2.18%	−0.09	8.32%	−0.12

Table 6: Performance metrics for the VAR model (m=1) under the Fixed Threshold Strategy.

We can notice that this model exhibits strong performance when targeting intraday returns, outperforming the index. However, when applied to standard log returns, the performance turns slightly negative



(a) Intraday trading.

(b) Whole day trading.

Figure 9: Cumulative strategy returns vs. NIFTY50 for the VAR model (m=1).

3.5 GARCH-in-Mean

Not satisfied with the previous results we try with a probabilistic approach. We decide to model the classical log returns $\log(\text{close}_{t+1}/\text{close}_t)$ (from now on r_{t+1}) using a GARCH-in-Mean process, where the conditional volatility enters the mean equation:

$$r_t = \mu + \lambda\,\sigma_t + \varepsilon_t, \qquad \sigma_t^2 = \omega + \alpha\,\varepsilon_{t-1}^2 + \beta\,\sigma_{t-1}^2,$$

where ε_t is white noise with zero mean and variance σ_t^2, and $\omega > 0$, $\alpha, \beta \geq 0$.

Given daily returns time series, we test the model from start to end of 2019 by fitting the it on a rolling window of length 50 days. For each time t, we:

(i) Compute residuals: $\varepsilon_{t-1} = r_{t-1} - \mu - \lambda\sigma_{t-1}$.

(ii) Update conditional variance: $\sigma_t^2 = \omega + \alpha\varepsilon_{t-1}^2 + \beta\sigma_{t-1}^2$.

(iii) Calculate forecast volatility: $\sigma_t = \sqrt{\sigma_t^2}$.

(iv) Forecast expected return: $\mathbb{E}[r_t \mid \mathcal{F}_{t-1}] = \mu + \lambda\sigma_t$.

To quantify uncertainty, for quantile level $q \in (0, 1)$, the q-quantile forecast return is

$$r_t^{(q)} = \mathbb{E}[r_t \mid \mathcal{F}_{t-1}] + z_q\sigma_t,$$

where $z_q = \Phi^{-1}(q)$. The corresponding price quantile forecast is

$$P_t^{(q)} = P_{t-1}\exp\left(r_t^{(q)}\right),$$

with P_{t-1} the most recent price.

Confidence Band	Return Coverage	Return Winkler
90% (0.050–0.950)	86.99%	0.0395
80% (0.100–0.900)	75.20%	0.0324

We can notice that the 90% and 80% predictive intervals exhibit slightly lower empirical coverage than their nominal levels, indicating mild undercoverage due to residuals normality assumptions failing on certain periods. The Winkler scores demonstrate that the model improves when using tighter intervals—although some data points fall outside these intervals, it does not significantly worsen the overall score. The forcasted variance looks consistent with the 20 days mean historical data.

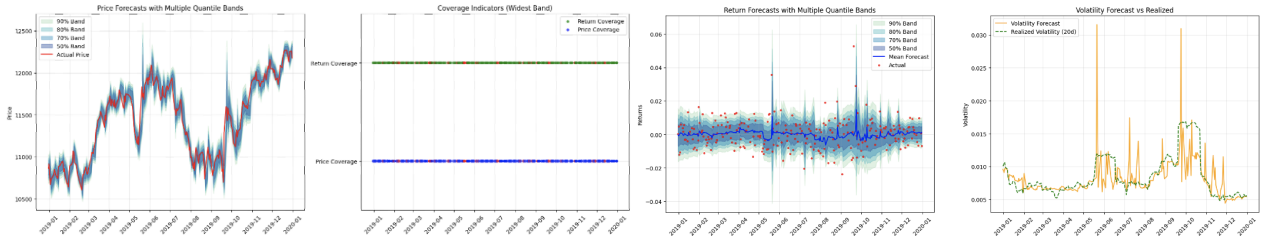


Figure 10: Quantile predictions

3.5.1 Tail-Based Mean Reversion Strategies

In order to exploit the probabilistic approach we implement two new custom trading strategies for our GARCH in-Mean quantile predictor model.

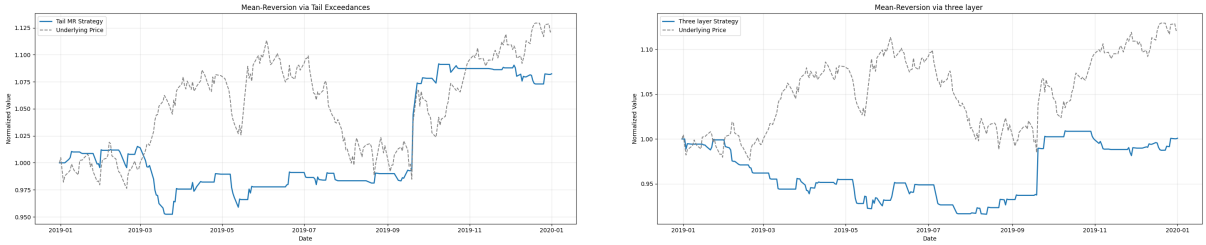
1. **Tail Exceedance Mean Reversion Strategy.**
   This strategy builds a signal based on the frequency of return breaches beyond fixed quantiles over a rolling window of 5 days. Specifically, it uses the 25th and 75th percentiles. A high number of downside exceedances suggests an overreaction and triggers a long position, while frequent upside exceedances indicate overextension and prompt a short position.

2. **Three-Layer Tail Mean Reversion Strategy.**
   This method uses a hierarchy of quantile bands: outer ($q_{0.10}$– $q_{0.90}$), middle ($q_{0.15}$–$q_{0.85}$), and inner ($q_{0.25}$–$q_{0.75}$). Positions are initiated when returns breach the outer bands and are held as long as returns remain within the middle or inner range. The position is exited if the return crosses the inner band in a favorable direction (profit), reverts within it (trend exhaustion), or breaches the opposite tail (regime shift).

In both cases, transaction costs of 5bp are accounted for when changing positions. The performance results of these two strategies applied to the log returns $\log(\text{close}_{t+1}/\text{close}_t)$ are shown in the following table and plots:

Table 7: Performance Metrics of Mean-Reversion Strategies

Strategy	Ann. Return	Sharpe	Max DD	Sortino
Tail Exceedance	8.42%	0.99	6.11%	1.57
Three-Layer Tail	-0.24%	0.01	8.19%	0.14



(a) Tail exceedance strategy.

(b) Three layer strategy.

Figure 11: Cumulative strategy returns vs. NIFTY50 for the GARCH-in-Mean model.

The Tail Exceedance Mean Reversion strategy significantly outperforms the Three-Layer Tail Mean Reversion across all key performance metrics. It achieves a notably higher annualized return (8.42% vs.-0.24%), along with superior risk-adjusted performance as reflected by its Sharpe (0.99 vs. 0.01) and Sortino (1.57 vs. 0.14) ratios. Moreover, it exhibits a lower maximum drawdown (6.11% vs. 8.19%), indicating reduced downside risk. Despite its strong performance metrics, the exceedance-counting strategy still fails to outperform the underlying index over the analyzed period.

4 Conclusions

Predicting financial markets using only past prices remains challenging, and our experiments reveal that model complexity does not always translate into better trading performance. The simple linear regression one-step-ahead strategy outperformed both its intraday counterpart and more elaborate Conv–LSTM, VAR and GARCH-in-Mean approaches. This suggests that, in the absence of external information, parsimony can be an asset and that increasing model sophistication without richer features may add noise rather than predictive power. Future work should therefore focus on integrating exogenous drivers and enhancing feature engineering and to assess in this setting whether additional model complexity and more advanced trading strategies can lead to improved performance.

References

[1] Zahra Fathali, Zahra Kodia Aouina, and Lamjed Ben Said. "Stock Market Prediction of NIFTY 50 Index Applying Machine Learning Techniques". In: *Applied Artificial Intelligence* 36 (2022). DOI: 10.1080/08839514.2022.2111134. URL: https://www.researchgate.net/publication/363632317_Stock_Market_Prediction_of_NIFTY_50_Index_Applying_Machine_Learning_Techniques.

[2] Gang Ji et al. "An adaptive feature selection schema using improved technical indicators for predicting stock price movements". In: *Expert Systems with Applications* 200 (2022), p. 116941. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2022.116941. URL: https://www.sciencedirect.com/science/article/pii/S0957417422003712.

[3] Fatemeh Moodi, Amir Jahangard-Rafsanjani, and Sajad Zarifzadeh. *Feature selection and regression methods for stock price prediction using technical indicators*. 2023. arXiv: 2310.09903 [q-fin.ST]. URL: https://arxiv.org/abs/2310.09903.

[4] Huiwen Wang, Wenyang Huang, and Shanshan Wang. *Forecasting open-high-low-close data contained in candlestick chart*. 2021. arXiv: 2104.00581 [econ.EM]. URL: https://arxiv.org/abs/2104.00581.