

Machine Learning-Assisted Validation of Supertrend Trading Signals Across Multiple Timeframes

Venkata Maheswara Rao Atchula¹, Dr. Pawan Kumar Pareek²

^{1,2}Nirwan University, Jaipur, Rajasthan

Abstract—Financial trading systems frequently combine deterministic technical indicators with machine learning in an attempt to reduce low-quality signals while retaining the interpretability of a rule-based strategy. This study evaluates a machine learning-assisted validation framework applied to Supertrend trading signals. The experimental design uses seven years of historical market data, with four years allocated to training/development and three years reserved for unseen machine-learning test evaluation. The experiment output contains rule-based training runs and machine-learning test runs across five timeframes 1-minute, 3-minute, 5-minute, 15-minute and 1-hour and evaluates two confidence thresholds explicitly represented in the supplied results, 0.50 and 0.53. The analysis records total orders, wins, losses, win percentage, total profit, total loss, net realized result and maximum drawdown. Across the training configurations, the descriptive sum of net realized results is -118,653.94, whereas the corresponding machine-learning test configurations sum to +3,992.86 across independent runs. Within the paired threshold tests, the higher 0.53 threshold reduces order count in every tested timeframe and reduces maximum drawdown in four of the five timeframe comparisons, while its effect on net realized performance is mixed. The results indicate that confidence-threshold changes alter signal selectivity and risk characteristics, but do not produce a monotonic relationship with profitability. The study is therefore framed as an empirical validation of threshold sensitivity and out-of-sample behaviour rather than as evidence of a universally optimal Supertrend configuration.

Index Terms—Supertrend, Machine Learning, Algorithmic Trading, Signal Validation, Technical Analysis, Confidence Threshold, Out-of-Sample Testing, Financial Markets.

I. INTRODUCTION

Algorithmic trading systems increasingly combine deterministic trading rules with statistical and machine-learning components. A rule-based indicator

can provide an interpretable mechanism for generating directional signals, while a machine-learning layer can be used as an additional validation stage. This architecture separates the generation of a candidate trade from the decision to permit that candidate to proceed.

Supertrend is an ATR-based trend-following indicator that combines trend direction with a volatility-sensitive band. Its commonly documented construction uses the midpoint of the high and low together with an ATR multiplier, with the active band switching when price crosses the relevant trend boundary. TradingView documents Supertrend as a trend-following indicator based on Average True Range and attributes its creation to Olivier Seban. [1] Machine-learning research in financial markets has used technical indicators, price-derived variables and other market information as predictors across classification and regression settings. A systematic review by Kumbure et al. examined 138 studies and found substantial variation in variables, markets, algorithms and validation methods. [2] Kelly and Xiu likewise emphasize that financial machine learning requires careful attention to the characteristics of financial data and the design of empirical evaluation. [3]

The present study focuses on a narrower operational question: whether a machine-learning validation layer changes the observed characteristics of candidate Supertrend signals when evaluated across multiple timeframes and confidence thresholds. The ML layer is treated as a filter rather than as a replacement for the underlying trading rule.

The experimental output data contains 15 runs. Five are identified as training configurations and ten as ML-test configurations. The ML-test configurations are paired at thresholds 0.50 and 0.53 across five timeframes. The paper uses these supplied results as

the empirical basis and does not reinterpret the run labels as a pooled portfolio.

1.1 Supertrend Signal Generation

The Supertrend indicator is designed to represent the prevailing directional state using an ATR-scaled trend boundary. In the common formulation, the basic upper and lower bands are constructed around the high-low midpoint and shifted by a multiple of ATR. The bands are then carried forward according to the trend state, and the active line changes when the relevant price-crossing condition is satisfied. [1]

In the experimental system examined here, the strategy is identified in the experimental output data as Supertrend and is evaluated at 1m, 3m, 5m, 15m and 1h intervals. The empirical analysis is restricted to the behaviour documented by the experiment results.

II. RESEARCH QUESTIONS AND OBJECTIVES

RQ1: How do the observed performance characteristics of Supertrend training configurations compare descriptively with the supplied ML-test configurations?

RQ2: How does changing the ML confidence threshold from 0.50 to 0.53 affect trade frequency across the tested timeframes?

RQ3: How does the threshold change affect net realized results and maximum drawdown across the paired timeframe experiments?

RQ4: Are the effects of threshold changes consistent across 1m, 3m, 5m, 15m and 1h configurations?

RQ5: What limitations must be considered before interpreting the historical results as evidence of general or live-market performance?

III. LITERATURE REVIEW

3.1 Supertrend and Volatility-Aware Trend Following
Supertrend belongs to the family of volatility-aware trend-following indicators. Its use of ATR makes the distance between price and the trend boundary responsive to measured volatility rather than fixed in absolute price units. TradingView's technical documentation describes the indicator as combining trend detection and volatility and provides the commonly used band and state-transition calculations. [1]

3.2 Machine Learning in Financial Market Analysis
Machine-learning methods have been widely investigated for financial forecasting and trading-related tasks. Kumbure et al. reviewed 138 journal articles published between 2000 and 2019 and reported broad use of technical indicators and multiple supervised and unsupervised learning methods. [2] Kelly and Xiu provide a broader survey of financial machine learning and emphasize the importance of adapting learning and evaluation procedures to financial data. [3]

3.3 Out-of-Sample Evaluation and Data-Snooping
Historical strategy evaluation is vulnerable to data-snooping and selection effects when many alternative configurations are tested. Sullivan, Timmermann and White developed a bootstrap-based framework for evaluating technical trading rules while accounting for data-snooping effects. [4] Bailey et al. further developed the concept of the probability of backtest overfitting and showed why repeated strategy selection on historical simulations can produce misleadingly strong in-sample results. [5]

3.4 Research Gap

The empirical question addressed here is not whether Supertrend is universally profitable, nor whether machine learning can forecast prices in general. Instead, the study examines the effect of placing a configurable ML confidence filter around an existing Supertrend signal process, with explicit comparison of two confidence thresholds and five timeframes over a seven-year train/test design. This provides a focused basis for studying selectivity, trade activity, net realized outcomes and drawdown.

IV. RESEARCH METHODOLOGY

The study follows a design-science-oriented artifact evaluation approach combined with historical computational experimentation and out-of-sample testing. The core architecture separates deterministic signal generation from machine-learning validation. The supplied results are treated as experiment-level observations rather than as a single combined trading account.

4.1 Methodological Pipeline

Stage	Function
1. Historical Market Data	Seven years of historical data
2. Supertrend Signal Generation	Rule-based candidate directional signals
3. ML Validation	Candidate signals evaluated by an ML validation layer
4. Confidence Threshold	Candidate accepted or filtered according to configured threshold
5. Trade Outcome	Accepted signals produce recorded orders and performance metrics
6. Performance Evaluation	Orders, wins/losses, win percentage, profit/loss, net realized and maximum drawdown

Table 1. Experimental processing pipeline.

4.2 Data Partition

Dataset role	Duration	Purpose
Training / Development	4 years	Strategy and ML development/training
Unseen ML Testing	3 years	Out-of-sample evaluation of ML-assisted configurations

Table 2. Seven-year train/test partition used for the study.

4.3 Experimental Dimensions

- Historical period: seven years, divided into four years of training/development and three years of unseen ML testing.
- Timeframes represented in experimental output data: 1m, 3m, 5m, 15m and 1h.

- ML confidence thresholds explicitly represented in the supplied ML-test configurations: 0.50 and 0.53.
- Training configurations: five runs, one for each tested timeframe.
- ML-test configurations: ten runs, forming threshold pairs at 0.50 and 0.53 for each timeframe.
- Performance measures: total orders, wins, losses, win percentage, total profit, total loss, net realized and maximum drawdown.

4.3.1 Supertrend Strategy Implementation

The supplied implementation defines Supertrend as an ATR-based trend-following strategy with Wilder-style RMA calculation, iterative upper and lower band smoothing, direction-state transitions, optional volatility filtering, moving-average trend gating, confirmation bars, and a downstream machine-learning validation stage. The implementation therefore provides a concrete description of the signal-generation and validation pipeline used to create the experimental signals, rather than treating Supertrend as a generic indicator.

The default Supertrend configuration in the supplied strategy code is an ATR period of 10 and a factor of 3.0. The implementation also supports ATR_RISING and ATR_MIN volatility filters, a configurable confirmation-bar count, and an MAEngine with a default disabled mode and 9/21 short/long lengths. These are implementation capabilities; experimental output data should be interpreted using the configuration actually used in each run.

Component	Implementation documented in supplied code
ATR	Wilder RMA; $\alpha = 1 / \text{ATR period}$
Default ATR period	10
Default Supertrend factor	3.0
Basic bands	$HL2 \pm \text{factor} \times \text{ATR}$
Band smoothing	Iterative upper/lower band carry-forward logic
Signal state	st_direction changes
BUY condition	Current Supertrend direction changes to -1
SELL condition	Current Supertrend direction changes to +1
Volatility filters	NONE, ATR_RISING, or ATR_MIN
Confirmation	Configurable confirm_bars
MA gate	MAEngine; default configuration disabled, 9/21 lengths
ML stage	MLValidatorClient validates generated feature dictionary before entry

Table 4. Supertrend strategy implementation used to generate candidate signals and validate entries.

4.3.2 Signal-Generation Logic

For each bar, the strategy computes true range from the current high-low range and the two previous-close gap components. ATR is then calculated using Wilder's recursive moving-average equivalent. Upper and lower basic bands are formed around HL2 using the configured factor. The implementation carries these bands forward according to prior band and prior-close conditions. A direction state is initialized and subsequently changes when price crosses the previous active band. A change in direction creates a candidate BUY or SELL signal.

Candidate signals are subsequently subjected to the configured trend and volatility gates. The supplied implementation contains a legacy trend-filter hook that currently returns true, while the MAEngine provides the active directional gate. If a volatility filter

is configured as ATR_RISING, the current ATR must exceed the previous ATR; with ATR_MIN, current ATR must meet the configured minimum. A confirmation-bar mechanism can additionally require repeated directional confirmation before entry.

4.3.3 Feature Construction for ML Validation

The supplied feature-builder implementation constructs a unified feature dictionary immediately before ML validation. For Supertrend, the strategy supplies Supertrend-specific state and engineered market context, then passes the complete dictionary to the ML validator. The feature builder contains conditional Renko-feature logic, but that branch is activated only when the strategy name contains 'renko'; therefore, Renko-specific features are not added to the Supertrend feature vector described here.

Feature group	Features / variables	Role
Strategy metadata	strategy, strategy_id, symbol, timeframe	Identifies the strategy and execution context
Trade context	direction, equity_before_trade, timestamp, entry_price	Captures direction and trade state at candidate entry
Supertrend state	supertrend, st_direction, distance_from_st, atr, factor	Represents indicator state and price distance from the active Supertrend
MA context	prev_short_ma, prev_long_ma	Provides short/long MA state from MAEngine
Base features	ma_slope, abs_ma_slope, market_regime, sl_points, tp_points, rr_ratio	Rule-independent baseline and risk descriptors
Engineered features	trend_strength, momentum, volatility, risk_pressure, reward_pressure, signal_quality, ma_slope, market_regime	Derived predictors used to enrich the ML validation input
ML output fields	ml_allowed, ml_score, ml_reason, ml_model, ml_prediction, ml_latency_ms	Validation result recorded after model evaluation

Table 5. Feature groups supplied to the machine-learning validation stage for the Supertrend strategy.

4.3.4 Base and Engineered Feature Definitions

The base-feature module calculates a 9-period and 21-period simple moving average from the recent price sequence (the feature-builder defaults) and defines `ma_slope` as `short_MA` minus `long_MA`. It also records the absolute slope, a trend/range label, stop-loss points, take-profit points and the reward-to-risk ratio. When the available price history is shorter than the long window, these base values fall back to zero/range defaults.

The feature-builder then derives additional predictors. `trend_strength` defaults to `ma_slope` divided by `brick_size` when `brick_size` is available; `momentum` defaults to `ma_slope` multiplied by absolute `ma_slope`; `volatility` is the range between the maximum and minimum of the most recent 20 prices when sufficient

history is available; `risk_pressure` and `reward_pressure` normalize the configured stop and target values; and `signal_quality` defaults to absolute MA slope multiplied by the reward-to-risk ratio. Runtime-provided values can override these defaults through the safe-override mechanism. For the Supertrend strategy, `brick_size` is not a Supertrend-specific feature and the Renko branch is not activated, so any brick-size-dependent default must be interpreted according to the actual runtime configuration.

4.3.5 ML Validation Flow

After a candidate signal survives the Supertrend, volatility and MA gates, the strategy constructs the feature dictionary and invokes the ML validator. The

returned validation record contains an allow/deny decision, score, reason, model identifier and prediction. The measured ML-validation latency is also recorded. An entry order is sent only when the validator returns an allowed result. This establishes the ML component as a post-signal validation layer rather than the primary generator of the Supertrend signal.

The supplied strategy also persists entry features and trade state and records MAE, MFE, holding time and exit reason for completed trades. Risk management is delegated to the RiskEngine, with SL/TP/TSL handling and signal-reversal exits represented in the exit payload. These execution-state fields are relevant to reproducibility and future research, although the present experimental output data reports the aggregate trading metrics rather than a feature-level attribution analysis.

4.4 ML Validation and Thresholding

The validation architecture treats the Supertrend strategy as the candidate-signal generator and the ML component as a decision filter. A candidate signal is evaluated by the ML layer, which produces a confidence value used by the configured threshold. At 0.50, candidates meeting the threshold are retained; at 0.53, the acceptance criterion is more selective

4.5 Evaluation Metrics

Metric	Definition / Role
Total Orders	Number of recorded orders in an experiment run
Wins / Losses	Counts reported by the experiment registry
Win Percentage	Win percentage reported by the source registry
Total Profit	Aggregate profitable trade amount reported by the registry
Total Loss	Aggregate losing trade amount reported by the registry
Net Realized	Total profit minus total loss, as reported
Maximum Drawdown	Largest reported drawdown for the experiment run

Table 3. Evaluation metrics used in the experimental output data.

V. EXPERIMENTAL RESULTS

The supplied Supertrend experiment contains 15 experiment rows: five training configurations and ten ML-test configurations. The rows represent independent strategy/timeframe configurations and should not be interpreted as a single pooled trading account. Descriptive sums are used only to summarize experiment coverage.

Group	Runs	Orders*	Net Realized*	Positive Runs	Largest Row DD
TRAIN	5	26,727	-118,653.94	3/5	101,723.25
ML-TEST	10	4,470	3,992.86	5/10	18,300.36

Table 4. Descriptive comparison of training and ML-test experiment rows. *Sums across independent runs; not a pooled portfolio.

The five training runs contain 26,727 recorded orders and have a descriptive net realized sum of -118,653.94. Three of the five training rows are positive, while the 1m and 3m training configurations are negative in the experimental output data. The largest maximum drawdown reported among the training rows is 101,723.25.

The ten ML-test runs contain 4,470 recorded orders and have a descriptive net realized sum of 3,992.86. Five of the ten ML-test rows are positive. The largest maximum drawdown among the ML-test rows is 18,300.36. These group-level differences are descriptive only because the groups correspond to different periods/configurations and do not constitute

a controlled causal comparison between two otherwise identical portfolios.

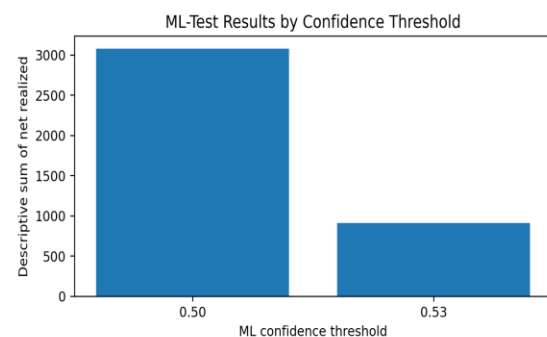


Figure 1. Descriptive sum of net realized across the supplied ML-test runs by confidence threshold.

5.1 Threshold Sensitivity

Threshold	Runs	Orders*	Net Realized*	Positive Runs	Largest Row DD
0.50	5	2,738	3,079.33	3/5	18,300.36
0.53	5	1,732	913.53	2/5	13,425.59

Table 7. ML-test threshold-level descriptive results. *Sums across independent runs.

The threshold results show a clear selectivity effect. Raising the threshold from 0.50 to 0.53 reduces the number of orders in every timeframe pair. The aggregate order count falls from 2,738 to 1,732 across the ten ML-test rows grouped by threshold. The descriptive net realized sum also changes from 3,079.33 at 0.50 to 913.53 at 0.53.

The profitability response is not monotonic. A higher threshold improves the net realized result in the 3m and 15m pairs, while it reduces the result in the 1m, 5m and 1h pairs. Thus, greater selectivity does not imply a uniform improvement in profitability.

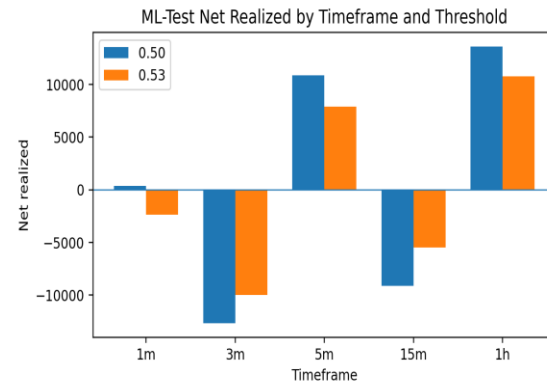


Figure 2. Net realized results for the paired ML-test thresholds across the five timeframes.

5.2 Timeframe-Level Observations

TF	Orders @0.50	Orders @0.53	Net @0.50	Net @0.53	DD @0.50	DD @0.53
1m	859	317	385.87	-2,341.17	4,582.87	5,211.88
3m	883	658	-12,684.14	-9,964.06	15,474.95	11,119.04
5m	655	492	10,861.04	7,892.58	5,605.62	4,293.84
15m	293	231	-9,107.44	-5,457.27	18,300.36	13,425.59
1h	48	34	13,624.00	10,783.45	4,957.85	4,957.85

Table 8. Paired threshold comparison by timeframe.

At 1h, both tested thresholds produce positive net realized results, while the 0.53 threshold reduces the number of orders from 48 to 34 and leaves the reported maximum drawdown unchanged at 4,957.85. At 5m, both thresholds are also positive, with fewer orders and lower drawdown at 0.53. At 1m, the net result changes from a small positive value at 0.50 to a negative value at 0.53. At 3m and 15m, both thresholds remain negative, but the 0.53 threshold reduces the magnitude of the negative net result and lowers maximum drawdown.

Across the five paired comparisons, the 0.53 threshold reduces maximum drawdown in the 3m, 5m and 15m configurations, while the 1m drawdown increases and the 1h drawdown remains unchanged. The observed results therefore support a selectivity-risk trade-off, but not a universal threshold rule.

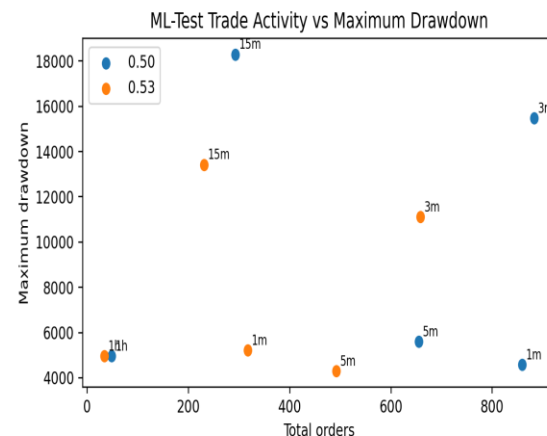


Figure 3. Relationship between trade activity and maximum drawdown across ML-test runs.

5.3 Training-Period Observations

Training TF	Orders	Win %	Net Realized	Max DD
1h	227	40.97%	16,997.63	18,187.38
15m	1,037	31.53%	12,679.03	11,355.89
5m	3,001	31.56%	1,408.64	17,852.01
3m	4,392	29.14%	-49,233.06	49,985.47
1m	18,070	28.46%	-100,506.18	101,723.25

Table 9. Supplied training-period Supertrend configurations.

The training configurations display substantial variation across timeframes. The 1h and 15m runs are positive, the 5m run is close to neutral but positive, and the 3m and 1m runs are negative. This variation is important because it shows that the baseline strategy itself is timeframe-sensitive before any ML-test threshold comparison is considered.

VI. DISCUSSION

The central empirical observation is that ML confidence thresholding changes signal selectivity. The 0.53 threshold accepts fewer candidate signals than 0.50 in every paired timeframe represented in the experimental output data. This reduction in activity is accompanied by lower maximum drawdown in several configurations, but the change in net realized result varies by timeframe.

The results also illustrate why threshold selection should not be based on trade frequency or drawdown alone. For example, the 15m configuration improves from -9,107.44 to -5,457.27 when the threshold increases from 0.50 to 0.53, while the 5m configuration declines from 10,861.04 to 7,892.58. The same threshold change therefore produces different economic effects under different timeframe conditions.

The seven-year design with four years of development data and three years of unseen testing is useful for separating model-development behaviour from later evaluation. However, the experimental output data contains only a limited number of thresholds and timeframe configurations. The observed results should therefore be treated as experimental evidence about the supplied configurations rather than as proof of a universally superior threshold.

The results are also consistent with established concerns about data-snooping and backtest overfitting. Sullivan et al. showed why technical-rule evaluation

should account for data-snooping, while Bailey et al. highlighted the risk of overfitting when investment strategies are selected from historical simulations. [4,5] The present study addresses part of this concern through an explicit unseen test period, but further statistical testing would be required for stronger claims.

VII. RESEARCH CONTRIBUTIONS

- A modular empirical framework in which a rule-based Supertrend strategy is evaluated together with a machine-learning validation layer.
- A seven-year experimental design with four years of training/development and three years of unseen ML testing.
- A multi-timeframe evaluation covering 1m, 3m, 5m, 15m and 1h configurations.
- A paired threshold-sensitivity analysis using the 0.50 and 0.53 confidence levels represented in the experimental output data.
- A performance evaluation combining trading activity, profitability and maximum drawdown.
- An explicit separation between descriptive experiment results and claims of general or live-market performance.

VIII. THREATS TO VALIDITY AND LIMITATIONS

- Historical backtesting: the experiments use historical data and do not establish live-market performance.
- Execution realism: experimental output data does not document a complete treatment of slippage, transaction costs, exchange latency, market impact or order-book effects.
- Configuration selection: multiple timeframes and thresholds create a multiple-comparison and

selection problem; observed historical outcomes should not be interpreted as proof of future performance.

- Metric reconciliation: some supplied rows have a small difference between total orders and the sum of reported wins and losses; the paper preserves the source metrics rather than reconstructing them.
- Independent runs: row-wise sums of net realized results are descriptive summaries across separate configurations, not a combined trading account.

IX. FUTURE RESEARCH

- Document the exact Supertrend parameterization, signal-generation rule and trade-management logic.
- Document the ML model architecture, feature set, target definition and probability-calibration method.
- Expand the confidence-threshold grid beyond the two levels represented in the experimental output data.
- Apply statistical tests or resampling procedures while accounting for multiple testing and data-snooping.
- Evaluate adaptive confidence thresholds that respond to volatility and market regime.
- Extend the experiment to additional assets and independent market periods.
- Introduce transaction costs, slippage and latency modelling.
- Conduct forward testing and live paper-trading validation before any real-capital deployment.

X. CONCLUSION

This study presents an empirical evaluation of machine-learning-assisted validation around Supertrend trading signals using a seven-year historical design with four years for training/development and three years for unseen testing. The experimental output data contains five training runs and ten ML-test runs across 1m, 3m, 5m, 15m and 1h timeframes, with ML confidence thresholds of 0.50 and 0.53 represented in the test configurations.

The ML-test configurations collectively contain 4,470 recorded orders and a descriptive net realized sum of 3,992.86, while the training configurations contain 26,727 orders and a descriptive net realized sum of -118,653.94. These figures are summaries across independent runs rather than a pooled account.

The paired threshold experiments show that increasing the confidence threshold from 0.50 to 0.53 consistently reduces trade activity, while the effect on maximum drawdown and profitability depends on timeframe. The evidence therefore supports interpreting the ML layer as a configurable selectivity mechanism rather than assuming that a higher confidence threshold is inherently better.

The principal value of the experiment is methodological: it demonstrates a reproducible way to study the interaction between a deterministic Supertrend signal process, ML confidence filtering, timeframe and out-of-sample performance. Future work should strengthen the statistical design, document the complete ML and strategy implementation, incorporate execution effects and validate the findings through additional assets and forward testing.

REFERENCES

- [1] TradingView, “Supertrend: Technical documentation describing the ATR-based Supertrend indicator, its calculation and origin attribution,” accessed Oct. 2026.
- [2] M. M. Kumbure, C. Lohrmann, P. Luukka, and J. Porras, “Machine learning techniques and data for stock market forecasting: A literature review,” *Expert Systems with Applications*, vol. 197, Art. no. 116659, 2022, doi: 10.1016/j.eswa.2022.116659.
- [3] B. T. Kelly and D. Xiu, “Financial machine learning,” *Foundations and Trends in Finance*, vol. 13, nos. 3–4, pp. 205–363, 2023, doi: 10.1561/05000000073.
- [4] R. Sullivan, A. Timmermann, and H. White, “Data-snooping, technical trading rule performance, and the bootstrap,” *The Journal of Finance*, vol. 54, no. 5, pp. 1647–1691, 1999, doi: 10.1111/0022-1082.00163.
- [5] D. H. Bailey, J. Borwein, M. López de Prado, and Q. J. Zhu, “The probability of backtest

overfitting,” *Journal of Computational Finance*, 2015, SSRN: 2326253.

- [6] J. W. Wilder, *New Concepts in Technical Trading Systems*. Trend Research, 1978.