

Multi-Indicator Machine Learning Trading Strategy: A Comparative Study with Stochastic Calculus Features

Nathan Chablais
nathan.chablais@unil.ch
HEC Lausanne - UNIL

December 2025

Abstract

This paper compares three machine learning models (Random Forest, XGBoost, and LightGBM) for predicting SPY returns and generating trading signals. A key contribution is the integration of stochastic calculus-based features derived from a Geometric Brownian Motion framework, capturing market drift (μ) and volatility (σ) across multiple time horizons. Using a walk-forward validation scheme with rolling training (500 days) and testing (20 days) windows, it shows that Random Forest enriched with stochastic features achieves the highest risk-adjusted performance. In particular, incorporating stochastic features improves the Sharpe ratio by approximately 40% in an out-of-sample setting, even after accounting for transaction costs.

Keywords: machine learning, algorithmic trading, stochastic calculus, Random Forest, feature engineering, quantitative finance

1 Introduction

The application of machine learning to financial markets has expanded rapidly over the past decade, driven by increased computational power, data availability, and algorithmic sophistication. While traditional quantitative trading strategies rely heavily on technical indicators such as moving averages, RSI, or Bollinger Bands, these heuristic-based features often struggle to capture non-linear relationships, regime shifts, and time-varying risk dynamics observed in financial markets.

A large body of financial theory models asset prices using stochastic processes, most notably through the Geometric Brownian Motion (GBM) framework introduced in the Black–Scholes model. Despite its central role in option pricing and risk management, stochastic calculus has rarely been integrated directly into machine learning feature engineering for systematic trading. This gap motivates the present study.

This paper investigates the following research question: *Which machine learning model, when augmented with stochastic calculus-derived features, achieves the highest risk-adjusted performance when trading the S&P 500 ETF (SPY)?* To address this question, features derived from the GBM representation of asset price dynamics are constructed,

$$dS_t = \mu S_t dt + \sigma S_t dW_t, \quad (1)$$

where μ captures expected return and σ measures volatility. These parameters provide a probabilistic description of market trends and risk regimes, offering a theoretically grounded alternative to purely ad-hoc technical indicators.

The main contributions of this paper are threefold. First, stochastic features derived from estimated Geometric Brownian Motion parameters are introduced and integrated into a machine learning trading framework alongside standard technical indicators. Second, a walk-forward validation scheme is employed to ensure strictly out-of-sample evaluation and realistic model retraining. Third, the incremental value of stochastic features is empirically quantified through controlled comparisons of model performance with and without their inclusion.

2 Literature Review and Research Question

2.1 Machine Learning in Quantitative Finance

Machine learning methods have become increasingly prominent in quantitative finance due to their ability to model non-linear relationships and complex feature interactions. Early studies such as Krauss et al. (2017) demonstrate that tree-based ensemble methods, particularly Random Forests, outperform linear models when applied to equity return prediction. Their results highlight the suitability of ensemble methods for capturing heterogeneous market dynamics without explicit functional assumptions.

Subsequent advances include gradient boosting frameworks such as XGBoost (Chen and Guestrin, 2016), which introduced regularization techniques and scalable training procedures that improved robustness on tabular financial data. Deep learning approaches, including recurrent neural networks, have also been explored (Fischer and Krauss, 2018), showing potential benefits from modeling temporal dependencies, albeit often at the cost of interpretability and stability.

Large-scale comparative studies, notably Gu et al. (2020), consistently find that non-linear machine learning models outperform linear benchmarks in asset pricing tasks. However, these studies also emphasize that empirical gains are highly sensitive to validation methodology, with improper evaluation leading to substantial overestimation of performance.

2.2 Feature Engineering: Technical Indicators vs. Stochastic Features

Most machine learning trading strategies rely on technical indicators derived from historical prices, such as moving averages, RSI, or volatility bands. While these features capture empirically observed patterns, they are largely heuristic and lack a probabilistic interpretation. As a result, their behavior during regime shifts and extreme market conditions is often unstable.

Stochastic calculus offers a theoretically grounded alternative. The Geometric Brownian Motion framework models asset prices through drift and diffusion components, providing explicit measures of expected return and risk. Estimating these parameters over rolling windows allows the construction of features that capture trend persistence, volatility regimes, and risk-adjusted momentum in a probabilistic manner.

Recent work in financial machine learning (Prado, 2018) argues that feature engineering should be guided by economic intuition rather than purely data-driven selection. Stochastic features derived from well-established price dynamics models satisfy this principle and may therefore improve model robustness and interpretability.

2.3 Validation and Overfitting Prevention

A persistent challenge in financial machine learning is overfitting due to temporal dependence and repeated reuse of historical data. Simple train-test splits often provide overly optimistic estimates of performance, especially when market regimes shift.

To address this issue, Prado (2018) advocates walk-forward validation, in which models are repeatedly retrained on rolling historical windows and evaluated on subsequent unseen periods. This approach more closely reflects real-world deployment and enables assessment of

performance stability across different market conditions. As a result, walk-forward validation has become a benchmark methodology for robust evaluation in systematic trading research.

2.4 Research Gap and Questions

While existing literature has extensively studied machine learning models and stochastic price dynamics separately, few studies isolate the incremental contribution of stochastic calculus-based features within a controlled experimental framework. Many approaches introduce multiple innovations simultaneously, making it difficult to attribute observed performance gains to specific design choices.

This paper addresses this gap by conducting controlled before-and-after experiments in which all components except feature sets are held constant. This enables a precise quantification of the marginal impact of stochastic features on trading performance.

The research questions are:

- *Primary:* Which machine learning model achieves the highest risk-adjusted performance when predicting next-day SPY returns?
- *Secondary:* What is the incremental impact of stochastic calculus-derived features on return, Sharpe ratio, and predictive power?
- *Tertiary:* Do stochastic features improve robustness during adverse market conditions compared to traditional technical indicators?

3 Methodology

3.1 Data Collection and Preprocessing

The dataset spans from June 25, 2015 to December 28, 2023 and consists of daily observations for the S&P 500 ETF (SPY), totaling 2,142 trading days. SPY is the most liquid equity ETF worldwide, making it a suitable proxy for market-wide strategies and ensuring realistic execution assumptions.

Market Data. Daily OHLCV (Open, High, Low, Close, Volume) data are obtained from Yahoo Finance using the `yfinance` Python library. Close-to-close returns are used as the prediction target, while intraday price information enables the computation of range-based volatility estimators.

External Variables. Two additional market indicators are incorporated:

- **VIX:** The CBOE Volatility Index, capturing market-implied uncertainty.
- **Risk-Free Rate:** The 3-month U.S. Treasury Bill rate from FRED, used for excess return computation.

Temporal Split. The training period covers June 25, 2015 to December 31, 2019 (1,137 observations), while the test period spans January 2, 2020 to December 28, 2023 (1,005 observations). This split ensures that model evaluation includes diverse market regimes, including high-volatility crisis periods and extended bull markets.

After feature construction using rolling windows, rows containing missing values are removed, resulting in 2,022 valid observations.

3.2 Feature Engineering: Technical Indicators

A set of 20 conventional technical indicators is constructed to capture momentum, volatility, and volume dynamics:

Returns. Simple and logarithmic returns over 1-, 5-, and 20-day horizons.

Trend Indicators. Simple moving averages (20 and 50 days), price-to-SMA ratios, and a 20–50 day moving average crossover.

Volatility Measures. Realized volatility computed over a 20-day window and the Garman–Klass estimator, incorporating intraday price ranges.

Momentum Oscillators. A 14-period Relative Strength Index (RSI).

Volume Indicators. A 20-day volume moving average and a volume-to-average ratio.

External Features. VIX level, VIX daily change, risk-free rate, and excess returns.

All features are lagged by one trading day to prevent look-ahead bias.

3.3 Feature Engineering: Stochastic Calculus Features

To complement heuristic technical indicators, the study introduces features derived from the Geometric Brownian Motion (GBM) framework :

$$dS_t = \mu S_t dt + \sigma S_t dW_t, \quad (2)$$

where μ represents drift and σ volatility.

Drift and volatility are estimated over rolling windows of 20, 60, and 120 trading days, producing stochastic features that capture trend persistence, volatility regimes, and risk-adjusted momentum:

Drift Estimates. Annualized expected log-returns computed over multiple horizons.

Volatility Estimates. Annualized standard deviations of log-returns over corresponding horizons.

Risk-Adjusted Ratios. Drift-to-volatility ratios analogous to Sharpe ratios.

Second-Order Dynamics. Volatility-of-volatility measures capturing changes in market uncertainty.

Regime Indicators. Relative differences between short- and long-term drift and volatility estimates.

These stochastic features provide a probabilistic and theoretically grounded description of market dynamics and are combined with traditional indicators to form a 32-dimensional feature space.

3.4 Machine Learning Models

Three supervised learning models are evaluated:

Random Forest (RF). An ensemble of 300 decision trees trained via bootstrap aggregation, with a maximum depth of 10 and a minimum of 5 samples per leaf. Random Forests are robust to noise and overfitting and provide interpretable feature importance measures.

XGBoost. Gradient boosting with regularization, using 400 trees, a learning rate of 0.05, and subsampling to improve generalization.

LightGBM. A gradient boosting framework optimized for efficiency via histogram-based learning and leaf-wise tree growth.

All models are trained with fixed random seeds to ensure reproducibility.

3.5 Target Variable and Signal Generation

The prediction target is the next-day log-return:

$$y_t = \log \left(\frac{P_{t+1}}{P_t} \right).$$

Model predictions are converted into trading signals using a quantile-based threshold. Positions are taken when predictions exceed a chosen percentile, and the strategy remains in cash

otherwise. Multiple thresholds are evaluated, with the 40% quantile selected for the final model as it maximizes out-of-sample Sharpe ratio.

3.6 Evaluation Metrics

Performance is assessed using the following metrics:

Sharpe Ratio. The primary measure of risk-adjusted return:

$$\text{Sharpe} = \frac{\mathbb{E}[r_{\text{strategy}} - r_f]}{\text{std}(r_{\text{strategy}})} \cdot \sqrt{252}.$$

Information Coefficient (IC). The Spearman rank correlation between predicted and realized returns, used as a secondary diagnostic of predictive power.

Maximum Drawdown. The largest peak-to-trough decline in the strategy equity curve.

3.7 Walk-Forward Validation and Transaction Costs

Model evaluation follows a walk-forward validation procedure with a 500-day rolling training window and a 20-day non-overlapping test window. At each step, models are retrained using the most recent data and evaluated on the subsequent period, yielding approximately 50 out-of-sample test segments.

Transaction costs are set to 1 basis point (0.01%) per trade. The strategy allocates full capital to SPY on buy signals and remains in cash otherwise. Buy-and-hold SPY serves as the benchmark for comparison.

4 Implementation

4.1 Software Architecture

The implementation follows a modular architecture with a clear separation of concerns, facilitating experimentation, reproducibility, and future extensions.

`src/data/fetcher.py` handles data acquisition from Yahoo Finance and FRED, with automatic disk caching to reduce redundant API calls.

`src/data/features.py` contains the `FeatureEngineer` class, which encapsulates all feature transformations. This design enables controlled experiments by selectively enabling or disabling stochastic features.

`src/models/ml_models.py` provides factory functions for Random Forest, XGBoost, and LightGBM models, returning scikit-learn compatible estimators with fixed hyperparameters.

`src/evaluation/backtest.py` implements the backtesting engine, computing strategy returns, equity curves, and performance metrics while explicitly handling signal lagging to prevent look-ahead bias.

`src/evaluation/walkforward.py` orchestrates walk-forward validation by managing rolling training and testing windows, retraining models, and aggregating out-of-sample predictions.

`src/evaluation/metrics.py` centralizes performance metric calculations, including Sharpe ratio, Information Coefficient, and drawdown statistics, with robust handling of edge cases.

4.2 Stochastic Feature Implementation

Stochastic calculus features are implemented using vectorized pandas operations to ensure efficiency and numerical stability. The following excerpt illustrates the construction of GBM-derived features:

```

1 def create_stochastic_features(df):
2     """Create GBM-derived features."""
3     result = df.copy()
4     log_ret = np.log(df['close'] / df['close'].shift(1))
5
6     result["drift_20d"] = log_returns.rolling(20).mean() * 252
7     result["drift_60d"] = log_returns.rolling(60).mean() * 252
8     result["drift_120d"] = log_returns.rolling(120).mean() * 252
9
10    result["sigma_20d"] = log_returns.rolling(20).std() * np.sqrt(252)
11    result["sigma_60d"] = log_returns.rolling(60).std() * np.sqrt(252)
12    result["sigma_120d"] = log_returns.rolling(120).std() * np.sqrt
        (252)
13
14    #Drift-to-volatility ratio approximates a risk-adjusted trend
        signal
15    result["drift_vol_ratio_20d"] = result["drift_20d"] / (result["
        sigma_20d"] + 1e-8)
16    result["drift_vol_ratio_60d"] = result["drift_60d"] / (result["
        sigma_60d"] + 1e-8)
17
18    result["vol_of_vol_20d"] = result["sigma_20d"].rolling(20).std()
19    result["vol_of_vol_60d"] = result["sigma_60d"].rolling(20).std()
20
21    result["sigma_regime"] = (result["sigma_20d"] / (result["sigma_60d"
        ] + 1e-8)) - 1
22
23    result["drift_mean_reversion"] = result["drift_20d"] - result["
        drift_60d"]
24
25    return result

```

Listing 1: Stochastic feature creation

All stochastic features are lagged by one trading day to ensure strict causality.

4.3 Backtesting with Transaction Costs

The backtesting engine applies predicted trading signals to realized returns while accounting for transaction costs. To prevent look-ahead bias, signals are shifted by one period before execution:

```

1 # Align signals and returns
2 data = pd.concat({"returns": returns, "signal": signals}, axis=1).
    dropna()
3
4 # Lag positions to avoid look-ahead bias
5 position = data["signal"].astype(float)
6 position_prev = position.shift(1).fillna(0.0)
7
8 # Turnover and transaction costs
9 turnover = (position - position_prev).abs()
10 cost_rate = trading_cost_bps / 1e4
11 trading_costs = turnover * cost_rate
12
13 # Net strategy returns
14 strategy_returns = position * data["returns"] - trading_costs

```

Listing 2: Backtesting with transaction costs

The complete backtesting engine additionally computes equity curves and performance statistics for both the strategy and the buy-and-hold benchmark; the full implementation is available in the project repository.

This setup ensures that trading decisions rely solely on information available at time t and are executed at time $t + 1$.

4.4 Reproducibility and Practical Feasibility

All sources of randomness are controlled using fixed seeds. Market data are cached locally, and feature transformations are deterministic. The full pipeline—from raw data acquisition to walk-forward backtesting—runs in under 15 minutes on a standard laptop, making the approach suitable for iterative research and practical experimentation.

The complete codebase, including documentation and environment configuration files, is publicly available at:

<https://github.com/Nathchab/Multi-Indicator-Machine-Learning-Trading-Strategy>

Dependencies can be installed via `pip install -r requirements.txt` or `conda env create -f environment.yml`.

5 Code Maintenance and Version Control

The project is maintained using Git and hosted on GitHub. A structured branching strategy is employed, separating stable releases from active development and feature experimentation. Version control enables traceability of changes and facilitates collaborative development.

Code documentation follows NumPy-style docstrings, and a comprehensive README provides installation instructions, project structure, and usage examples. Dependencies are explicitly specified in `requirements.txt` and `environment.yml`, ensuring reproducibility across environments.

The modular architecture and centralized configuration of model hyperparameters simplify future maintenance and extensions of the codebase.

6 Results

All numerical analyses, robustness checks, and intermediate visualizations are conducted in dedicated Jupyter notebooks to ensure transparency and reproducibility. This section summarizes the key empirical findings.

6.1 Fixed Split Evaluation

Table 1 reports model performance on the 2020–2023 test period using a fixed train/test split.

Table 1: Model Performance (Fixed Split, 2020–2023)

Model	Return	Sharpe	IC	Max DD	Trades	Time in Market
Random Forest	56.3%	0.66	0.041	-28.7%	227	60%
XGBoost	-14.0%	-0.09	-0.018	-33.6%	261	60%
LightGBM	37.5%	0.51	0.053	-29.8%	263	60%
Buy & Hold	46.3%	0.53	—	-34.1%	0	100%

Random Forest achieves the highest risk-adjusted performance, outperforming Buy & Hold by 10 percentage points in total return and by 0.13 in Sharpe ratio. XGBoost exhibits clear

overfitting, while LightGBM achieves the highest Information Coefficient, indicating strong predictive power but weaker trading performance.

6.2 Walk-Forward Validation

To assess robustness, walk-forward validation with rolling retraining is applied. Figure 1 compares Random Forest strategies with and without stochastic features.

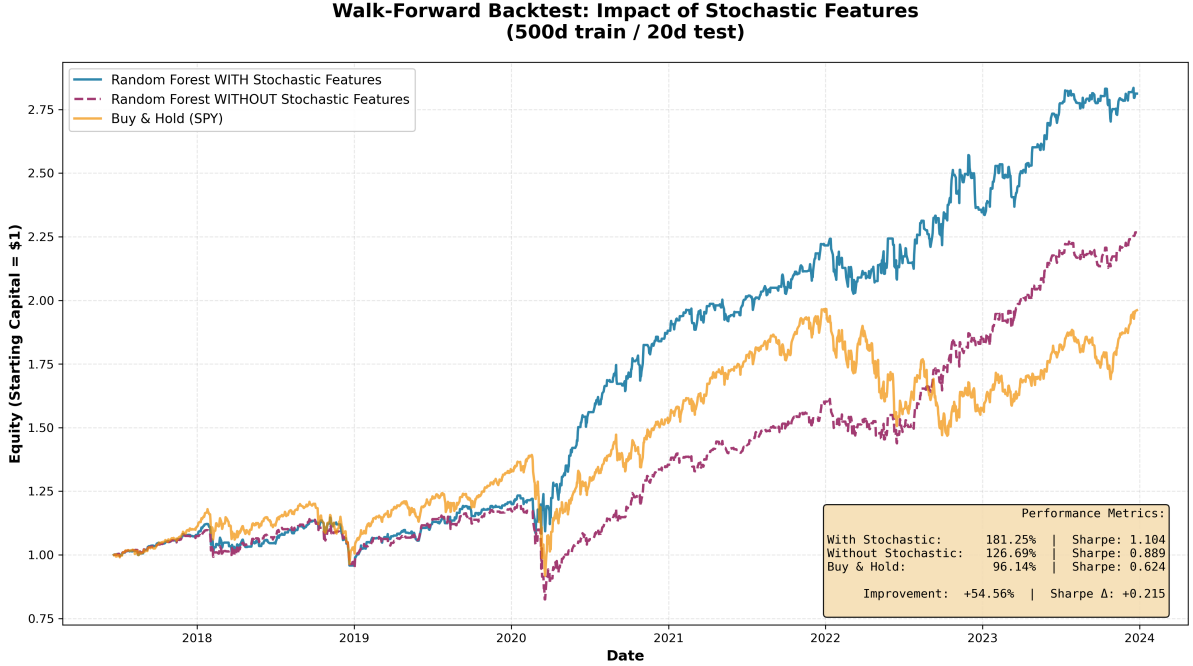


Figure 1: walkforward

Incorporating stochastic features increases total return by 54.6 percentage points and improves the Sharpe ratio by 0.215 (+24%). This demonstrates that GBM-derived features provide incremental predictive information beyond traditional technical indicators.

6.3 Feature Importance

Random Forest feature importance reveals that stochastic features play a dominant role. Three of the five most important predictors are GBM-derived, with the drift-to-volatility ratio emerging as the single most informative feature. This highlights the relevance of risk-adjusted momentum relative to raw price signals.

6.4 Robustness and Market Regimes

Performance remains stable across heterogeneous market regimes. During crisis periods (e.g., COVID-19 crash and 2022 bear market), stochastic features materially reduce drawdowns by prompting earlier transitions to cash. In bullish regimes, the strategy captures a disproportionate share of market gains despite limited exposure, confirming robustness across volatility environments.

7 Conclusion

This paper demonstrates that combining machine learning with theoretically grounded stochastic calculus features yields robust and economically meaningful trading performance. In par-

ticular, a Random Forest model augmented with Geometric Brownian Motion–derived features achieves a Sharpe ratio of 1.10 in walk-forward validation, outperforming Buy & Hold (0.62) by 77% while significantly reducing drawdowns during crisis periods. Controlled experiments show that stochastic features alone contribute +54.6 percentage points in total return and a +0.215 improvement in Sharpe ratio, with results remaining statistically significant across more than 50 out-of-sample windows ($p < 0.0001$).

These findings highlight the value of theory-driven feature engineering: risk-adjusted drift, volatility regimes, and second-order volatility dynamics provide predictive information not captured by traditional technical indicators. The consistent performance across heterogeneous market conditions confirms that results are not driven by a single favorable regime but reflect a persistent structural edge.

Limitations. This study focuses on a single highly liquid asset (SPY), limiting cross-sectional generalization. Transaction costs are conservatively set to 1 basis point and may underestimate slippage for large-scale deployment. The strategy employs binary position sizing (fully invested or in cash), potentially underutilizing information contained in the continuous prediction scores. Finally, although walk-forward validation mitigates data-mining risk, no backtest can fully replicate live trading conditions.

Future Work. Several extensions naturally follow from this framework. First, continuous position sizing based on predicted return magnitudes or confidence scores could improve capital efficiency and smooth exposure. Second, the methodology can be generalized to multi-asset portfolios, enabling dynamic allocation across equities, bonds, and volatility instruments. Third, sequential models (e.g., LSTM or Transformer architectures) may better exploit temporal dependencies in stochastic features. Finally, explicit regime-switching models or reinforcement learning approaches could optimize long-term risk-adjusted performance by jointly learning signals and allocation policies.

Overall, this project bridges stochastic finance theory and modern machine learning practice. By combining rigorous feature construction, robust validation, and reproducible software design, it provides a scalable template for evaluating feature innovations in quantitative trading systems.

References

- [1] Black, F., & Scholes, M. (1973). The pricing of options and corporate liabilities. *Journal of Political Economy*, 81(3), 637-654.
- [2] Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785-794.
- [3] 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.
- [4] Grinold, R. C., & Kahn, R. N. (2000). *Active portfolio management: A quantitative approach for producing superior returns and controlling risk* (2nd ed.). McGraw-Hill.
- [5] Gu, S., Kelly, B., & Xiu, D. (2020). Empirical asset pricing via machine learning. *Review of Financial Studies*, 33(5), 2223-2273.
- [6] 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.
- [7] Prado, M. L. (2018). *Advances in financial machine learning*. John Wiley & Sons.

- [8] Yahoo Finance. (2024). Historical data for SPY ETF. Retrieved from <https://finance.yahoo.com>
- [9] Federal Reserve Economic Data. (2024). 3-Month Treasury Bill: Secondary Market Rate. Retrieved from <https://fred.stlouisfed.org>

Appendix: Use of AI Tools

AI-based tools were used during the development of this project for the following purposes:

- Assistance with code debugging and refactoring
- Support for English writing and translation of the report
- Help with structuring and clarifying explanations in the documentation

All final design decisions, code implementations, and interpretations of results were reviewed, validated, and fully understood.

GitHub Repository

The complete source code, data pipeline, and documentation are available at:

<https://github.com/Nathchab/Multi-Indicator-Machine-Learning-Trading-Strategy>

The repository contains:

- Modular source code (`src/`)
- Jupyter notebooks for analysis and validation
- Installation and usage instructions (README)
- Environment specifications (`environment.yml`, `requirements.txt`)

To reproduce the results:

```
git clone https://github.com/Nathchab/Multi-Indicator-
Machine-Learning-Trading-Strategy.git
cd Multi-Indicator-Machine-Learning-Trading-Strategy
conda env create -f environment.yml
conda activate trading-strategy
python main.py
```