Chilean Journal of Statistics
Vol. 16, No. 2, December 2025, 178–216
DOI: 10.32372/ChJS.16-02-03

TIME-SERIES MODELS
RESEARCH ARTICLE

Short-horizon volatility spike forecasting via integrated functional and topological representations

ÇAĞLAR SÖZEN[1,*]

[1]Department of Finance and Banking, Görele School of Applied Sciences, Giresun University, Giresun, Turkey

(Received: 31 October 2025 · Accepted: 25 November 2025)

Abstract

We study early warning of short-horizon volatility spikes by combining three complementary blocks of information: functional data analysis summaries based on functional principal component analysis scores and derivative norms, multi-scale topological data analysis descriptors derived from persistent homology, and heterogeneous autoregressive lagged-volatility features. All models are trained under a strict chronological design, with operating thresholds pre-specified from out-of-fold scores to ensure that model selection and evaluation remain separate. Statistical significance is assessed using paired DeLong tests for the receiver operating characteristic area under the curve and is complemented by a calendar block bootstrap that more directly accounts for temporal dependence in the scores. Across four large-capitalization equities (INTC, COP, NVDA, DIS), the combined specification (BOTH, defined as functional data analysis features plus topological data analysis features plus heterogeneous autoregressive features) generally outperforms the functional data analysis-only and topological data analysis-only baselines in receiver operating characteristic area under the curve and precision–recall area under the curve, and improves their probabilistic accuracy and calibration while remaining competitive with a heterogeneous autoregressive-only gradient-boosted tree benchmark. In a regime-switching generalized autoregressive conditional heteroskedasticity stress experiment, a heterogeneous autoregressive-based gradient-boosted tree model attains the highest discrimination, illustrating that the empirical gains from combining functional and topological information are not guaranteed when the data-generating mechanism is close to a low-dimensional lag-volatility structure. Methodologically, we propose a time-respecting method that integrates functional and topological geometry with paired inference under temporal dependence. Practically, we provide a reproducible script for early warning and for monitoring short-run volatility stress.

Keywords: functional data analysis · persistent homology · random forest · topological data analysis · volatility spikes

Mathematics Subject Classification: Primary 62M10 · Secondary 62P05

*Corresponding author. Email: caglar.sozen@giresun.edu.tr (C. Sözen)

ISSN: 0718-7912 (print)/ISSN 0718-7920 (online)

http://www.soche.cl/chjs

1. Introduction

Short-horizon volatility spikes materially affect trading and risk management: they amplify execution costs and slippage, tighten or widen spreads, and trigger rapid adjustments in inventory, margins, and hedges for dealers, market makers, and clearing members alike.

Traditional conditional-variance and realized volatility (RV) frameworks provide tools for measurement and forecasting, yet the sharp, nonlinear, and time-varying patterns that accompany regime breaks often escape single-family representations [1, 2]. Our working premise is that short-horizon spike risk is inherently dual: it is at once a problem of functional shape—the local level, slope, and curvature of recent return paths—and of state-space geometry—the multi-scale connectivity and loop structure induced by the dynamics.

To address this, we jointly exploit two complementary feature families. From rolling windows of returns we extract functional summaries based on functional principal component analysis (FPCA) scores and discrete L^2 norms of first and second differences that encode smooth aspects of the path. In parallel, we construct topological summaries from delay-embedded absolute returns via persistent homology (PH), using Vietoris–Rips (VR) filtrations to characterize loop richness and multi-scale connectivity in the reconstructed state space [3, 4, 5, 6]. Stability results for persistence diagrams imply that small perturbations of the underlying paths lead to small changes in the associated diagrams, and in practice the scalar summaries we extract tend to be robust to moderate measurement noise. We then combine the functional and topological blocks—augmented with simple heterogeneous autoregressive (HAR) lags of volatility proxies—within a random forest (RF) classifier, which captures nonlinear interactions without imposing a parametric functional form [7].

We study whether an integrated feature set, denoted BOTH, that concatenates functional data analysis (FDA) features, topological data analysis (TDA) features, and HAR lags improves discrimination and decision quality relative to FDA-only and TDA-only baselines under time-aware validation on liquid single-name equities. In particular, we ask whether BOTH delivers higher receiver operating characteristic area under the curve (ROC–AUC) and precision–recall area under the curve (PR–AUC), and better-calibrated probabilities, when models are trained and tested under a strict chronological split.

We consider four liquid United States equities (INTC, COP, NVDA, DIS) over 2010–2024 and construct rolling, overlapping windows of daily log returns. For each window we compute FDA features (FPCA scores and derivative-based roughness measures) and TDA features (such as Betti-1 intensity, total persistence, and persistence entropy) from VR filtrations of the delay-embedded series [8, 9]. Models are trained and evaluated under strict chronology (train $\rightarrow$ calibration $\rightarrow$ test), with all preprocessing steps, feature transformations, and probability thresholds estimated on the training split and then applied forward in time. We report ROC–AUC and PR–AUC, summarize operating points via precision, recall, and F1 at pre-specified thresholds, and assess ROC–AUC differences using paired DeLong tests with calendar block-bootstrap contrasts [10]. Probability calibration is evaluated using the Brier score and expected calibration error (ECE). Lower values of both metrics indicate better alignment between predicted probabilities and observed spike frequencies; ECE is computed from $B_{\text{bin}} = 10$ equal-frequency bins whose cutpoints are defined once on the out-of-fold training probabilities and then kept fixed across models and on the test set. This article makes the following four contributions:

(i) We develop a compact, reproducible method that jointly leverages functional shape and topological geometry, with HAR-style volatility lags to encode short-memory structure. The feature construction is explicitly time-ordered and uses training-only winsorization, scaling, and embedding choices, which are then frozen and applied to later folds and to the test set.

(ii) We pair rolling-origin model selection with a strict, non-leaky train–calibration–test design, and we report both ranking metrics (ROC–AUC, PR–AUC) and probabilistic accuracy and calibration metrics (Brier score, ECE). This design emphasizes decision relevance under pronounced class imbalance and time-varying spike prevalence.
(iii) We complement DeLong's paired ROC–AUC tests with block-bootstrap contrasts that explicitly acknowledge serial dependence and overlapping windows. This dual approach yields a more realistic assessment of uncertainty for financial return series, where independent and identically distributed assumptions are rarely tenable.
(iv) On INTC, COP, NVDA, and DIS, the integrated specification BOTH typically achieves higher ROC–AUC and PR–AUC than the FDA-only and TDA-only baselines and offers better probability calibration. Relative to the HAR-only XGBoost benchmark, BOTH is clearly ahead on COP, roughly on par on NVDA, and somewhat behind on INTC and DIS, underscoring that the relative ranking of the two nonlinear approaches is asset-dependent and that calibration diagnostics should be interpreted alongside ranking metrics for downstream risk-based decisions.

The remainder of this article is organized as follows. Section 2 formalizes the prediction target, feature construction (FDA/TDA/HAR), learning objective, and time-aware inference procedures. In Section 3, we detail the empirical universe, preprocessing rules, window and label construction, and the cross-validation protocol. In Section 4, the main empirical findings are presented, including discrimination, paired ROC–AUC comparisons, calibration diagnostics, and cross-validation summaries. Section 5 reports a controlled stress experiment based on a regime-switching generalized autoregressive conditional heteroskedasticity (GARCH) data-generating process that mirrors volatility clustering and regime shifts. In Section 6, we combine discussion and conclusion, drawing out operational implications, limitations, and avenues for future work.

2. Methodology

This section formalizes the prediction target, feature construction based on FDA, TDA, and heterogeneous autoregressive (HAR) features, model families, time-aware validation, and statistical inference. The overarching principle is strict chronological design: all transformations and thresholds are pre-specified on the training split and then applied unchanged to later folds and the test set, preventing any form of look-ahead leakage.

2.1 *Related work*

Realized volatility (RV) offers a consistent nonparametric benchmark for integrated volatility and underpins much of the modern literature on volatility measurement and forecasting [1, 2]. In parallel, heavy-tailed parametric models for financial returns, including alpha-stable specifications, provide flexible tools for capturing tail risk [11]. In our empirical setting we work with a lower-frequency proxy based on averages of absolute daily returns, rather than the high-frequency realized variance used in these references. Periodic and regime-dependent forms of conditional heteroskedasticity have also been shown to improve volatility modelling in settings with structural variation over time [12].

FDA treats trajectories as smooth curves and summarizes their shape via FPCA and derivative norms. Expositions and finance-adjacent applications are given in [13, 14], with functional forecasting exemplars in [15, 16, 17]. Recent FDA developments also highlight the benefits of nonparametric functional regression for time-dependent predictive tasks, including in economic and financial applications [18]. Complementary robust multivariate procedures based on projection-type ideas have also been proposed; see, for example, [19] for a Stahel–Donoho estimator built from skewness-based random projection directions.

On the topological side, delay embedding reconstructs dynamical systems under broad conditions [3, 4]. PH then summarizes multi-scale connectivity and loop structure with stability guarantees [5, 6], and mature software ecosystems such as Ripser, GUDHI, and giotto-tda have made these methods computationally accessible [20, 21, 22, 23]. To turn persistence diagrams into finite-dimensional feature vectors, persistence landscapes, persistence images, and kernel-based constructions are standard approaches [24, 25, 26, 27, 28]. Empirically, TDA indicators have shown early-warning potential around extreme events in financial markets and macro-financial systems [29, 30]. Within this toolbox, we focus on scalar summaries such as Betti-1 intensity, total persistence, and persistence entropy because they are computationally light, empirically stable to small perturbations, and interpretable as measures of loop richness and multi-scale topological energy, providing complementary signal to smooth functional shape descriptors.

From an evaluation perspective, pronounced class imbalance and time-varying spike prevalence call for metrics beyond ROC–AUC alone. PR–AUC and the F1 score directly reflect decision quality when the positive class is rare or shifting, whereas ROC–AUC primarily measures ranking performance. For formal pairwise discrimination comparisons we rely on paired DeLong tests based on common evaluation dates, and we complement these with calendar block-bootstrap contrasts that explicitly acknowledge temporal dependence and overlapping windows [10, 31, 32]. Given the pronounced and time-varying spike prevalence documented in Table 2, we report both ranking metrics (ROC–AUC) and precision–recall measures (PR–AUC and F1), which are sensitive to class imbalance. We complement these with probabilistic accuracy and calibration metrics such as the Brier score and ECE, providing a fuller assessment of spike-risk forecasts across different prevalence regimes and operating points.

2.2 *Problem and target*

We work with daily adjusted closing prices $\{P_t\}$ for a given asset. Daily log returns are defined by $r_t = \log(P_t) - \log(P_{t-1})$, which provide a standard, scale-invariant proxy for percentage moves. To construct predictors that reflect recent path behavior, we use overlapping windows of fixed length $W = 60$ trading days and write $X_i = (r_{i-W+1}, \ldots, r_i)$ for the ith window. This representation ensures that only information available up to day i is used for forecasting.

The prediction target is a short-horizon volatility spike indicator. As a forward-looking volatility proxy, we use the K-day mean absolute return over the subsequent period ($K = 5$) expressed as $\mathrm{RV}_{i\to i+K} = (1/K)\sum_{j=1}^{K} |r_{i+j}|$, which captures the average level of realized volatility in the next trading week. Following the realized volatility literature we denote this proxy by RV, but note that, unlike high-frequency realized variance based on intraday squared returns [1, 2], it is a low-frequency average of absolute daily returns. Spike labels are defined on the training period by a train-only quantile rule as $Y_i = \mathbf{1}\{\mathrm{RV}_{i\to i+K} > \mathrm{Quantile}_{0.70}(\mathrm{RV}_{\mathrm{train}})\}$, so that the 70th percentile of the training distribution marks the onset of a volatility spike. This choice produces a nontrivial yet imbalanced classification problem and is fixed once and for all on the training split; class prevalence on the test set is therefore an outcome of the data rather than a design choice, which is important for assessing robustness under distributional drift.

Winsorization of returns uses the training quantiles $(q_{0.01}, q_{0.99})$ to mitigate the influence of extreme outliers while retaining heavy-tailed behavior. All centering and scaling parameters, principal component loadings, embedding delays, and filtration scales are fitted on the training split and then applied unchanged to subsequent folds and to the test set. This emulates a realistic forecasting environment in which model components are frozen once deployed.

2.3 *Feature engineering*

For each window X_i, we construct a feature vector $z_i \in \mathrm{R}^p$ that combines functional summaries of the recent path, topological information about the reconstructed state space, and simple lag-volatility cues $\phi_{\mathrm{FDA}}\colon X_i \mapsto \mathrm{R}^{p_f}$, $\phi_{\mathrm{TDA}}\colon X_i \mapsto \mathrm{R}^{p_t}$, $\phi_{\mathrm{HAR}}\colon X_i \mapsto \mathrm{R}^3$, and $z_i = (\phi_{\mathrm{FDA}}(X_i), \phi_{\mathrm{TDA}}(X_i), \phi_{\mathrm{HAR}}(X_i))$. This construction is designed to capture complementary aspects of spike risk: smooth path shape, nonlinear geometric structure, and low-frequency volatility dynamics.

On the training window matrix (rows: windows; columns: within-window time indices), we perform global principal component analysis and retain the first four scores, denoted `fscore1`–`fscore4`. These FPCA scores summarize the dominant modes of variation in recent return trajectories and can be interpreted as low-dimensional coordinates of path shape (level, slope, and curvature components). To further quantify within-window roughness and curvature, we compute discrete L^2 norms of first and second differences stated as $\|\Delta X_i\|_2 = (\sum_{t=2}^{W}(x_t - x_{t-1})^2)^{1/2}$ and $\|\Delta^2 X_i\|_2 = (\sum_{t=3}^{W}(x_t - 2x_{t-1} + x_{t-2})^2)^{1/2}$, where x_t denotes the tth return within window X_i. Large values of $\|\Delta X_i\|_2$ and $\|\Delta^2 X_i\|_2$ indicate more jagged and locally curved paths, which are natural precursors of turbulence.

Standardization based on z-scores is fitted on the training split (for all FDA features) and applied forward in time. This keeps the scale of features stable and avoids contaminating standardization with information from the test period.

While FDA captures smooth shape, it does not directly encode the geometry of the underlying dynamics. To this end, we use delay embedding of the absolute returns $|r_t|$ with embedding dimension $m = 3$. The delay τ is chosen on the training data as the first local minimum of the autocorrelation function of $|r_t|$, a traditional choice to avoid redundant coordinates while preserving dynamical information. The resulting point cloud in R^3 is given by $(y_t, y_{t-\tau}, y_{t-2\tau})$, which reconstructs the local state space geometry under mild conditions.

On this cloud, we run a VR filtration on an equispaced grid of 40 radius levels over $\varepsilon \in (0, \varepsilon_{\max})$, with $\varepsilon_{\max} = 0.6 \times \mathrm{median}\{\|x_a - x_b\|_2\colon a < b\}$, being computed on the embedded cloud in the training split. This choice of $\varepsilon_{\max}$ scales the filtration to the typical inter-point distances observed in the data and avoids numerically degenerate regimes with either no connectivity or a fully connected complex.

From the resulting persistence diagrams, we extract scalar summaries $T_{H_1}^{(1)}$, $T_{H_1}^{(2)}$, $T_{H_0}^{(1)}$, and E_{pers}, B_1, P_{tot}, where $T_{H_1}^{(1)}$ and $T_{H_1}^{(2)}$ denote the largest and second-largest lifetimes in homology group H_1, $T_{H_0}^{(1)}$ is the largest lifetime in H_0, $E_{\mathrm{pers}} = -\sum_i p_i \log(p_i)$ is persistence entropy based on normalized lifetimes p_i, B_1 summarizes Betti-1 intensity (loop richness), and $P_{\mathrm{tot}} = \sum_i \ell_i$ is total persistence (a multi-scale measure of topological energy). These statistics are designed to be computationally light, stable to small perturbations, and interpretable in terms of regime geometry. The 40-level VR grid provides a practical balance between empirical stability of these summaries and computational cost in rolling evaluations. All embedding and winsorization settings are reported in Table 1.

Table 1.: Embedding and winsorization summary

Asset	m	τ	Lower bound	Upper bound	Spike threshold
INTC	3	5	−0.0453001	0.0479312	0.0134972
COP	3	2	−0.0555492	0.0521633	0.0149908
NVDA	3	6	−0.0692662	0.0700310	0.0205731
DIS	3	5	−0.0415187	0.0422795	0.0109688

To encode coarse volatility memory, we include HAR-type averages of $|r_t|$: a short (5-day) average $h_i^{(d)}$, an intermediate (22-day) average $h_i^{(w)}$, and a longer (66-day) average $h_i^{(m)}$. These three scales provide a simple weekly, monthly, and quarterly-style multi-scale structure inspired by heterogeneous autoregressive models [33], but they are not an exact implementation of the original daily, weekly, and monthly HAR realized volatility specification. We benchmark three main feature sets: FDA, TDA, and the combined specification BOTH introduced in Section 1, which concatenates all feature blocks plus HAR lags.

2.4 *Learning objective, tuning, and operating points*

Given features z_i and labels $Y_i \in \{0, 1\}$, the learning objective is to approximate the conditional spike probability $\mathrm{P}(Y_i = 1 \mid z_i)$ and to derive decision rules with desirable discrimination and calibration properties.

The main classifier is an RF [7], chosen for its ability to capture nonlinear interactions and heterogeneous effects without strong parametric assumptions and for its robustness under moderately high-dimensional feature sets. We use rolling-origin five-fold cross-validation (CV) with ROC–AUC as the selection criterion. Hyperparameters are restricted to a low-dimensional pre-specified grid of mtry values given by $\mathrm{mtry} \in \left\{\lfloor\sqrt{p}\rfloor, \lfloor p/3\rfloor, \lfloor p/2\rfloor\right\}$, with each candidate truncated below at one, where p is the dimensionality of the feature set under consideration. All remaining RF hyperparameters are held fixed across assets and feature families: forests are fitted with $B = 500$ trees and `nodesize`$= 10$ during cross-validation, and then refitted with $B = 1000$ trees and the same node size on the full training split. For each model family, the value of mtry that maximizes mean cross-validated ROC–AUC across the five folds is selected and then held fixed when refitting on the full training segment and when constructing out-of-fold (OOF) probabilities for threshold selection.

Operating thresholds are determined only from OOF probabilities and then frozen prior to any test-set evaluation. We consider two scalar targets defined as

$$\text{Youden } J(\tau) = \mathrm{TPR}(\tau) + \mathrm{TNR}(\tau) - 1, \qquad F_1(\tau) = 2\left(\frac{\mathrm{Prec}(\tau)\,\mathrm{Rec}(\tau)}{\mathrm{Prec}(\tau) + \mathrm{Rec}(\tau)}\right),$$

where TPR denotes the true positive rate, TNR the true negative rate, Prec precision, and Rec recall. We select the Youden-J-optimal and F_1-optimal thresholds on the OOF scores. Using OOF scores ensures that threshold selection is based on data not used for fitting the corresponding trees, which protects against overfitting at the decision-rule level. For deployment, we treat the F_1-optimal OOF threshold as a single, pre-specified policy level targeting a balanced precision–recall trade-off; alternative operating points can be obtained from the same calibration split without retraining.

2.5 *Time-aware validation and leakage controls*

Financial time series exhibit serial dependence, structural breaks, and distributional drift. Randomly shuffling observations across time would therefore yield overly optimistic performance estimates and leak future information into the training process. To avoid this, all validation procedures respect calendar order.

All transformations and thresholds introduced in Sections 2.2 and 2.3 are always fitted using only past data. Within each cross-validation fold, they are estimated on the training block of that fold and then applied unchanged to the immediately subsequent validation block, preserving temporal causality. After cross-validation has selected the hyperparameters, we refit all transformations and models on the full training segment; these fitted objects are then frozen and applied unchanged to the test set.

Cross-validation folds follow a rolling-origin design: each fold trains on an initial segment of the series and validates on the immediately subsequent block, mimicking repeated train-then-deploy cycles. Predictions from competing models are aligned on identical dates, enabling valid paired comparisons and ensuring that differences between models are not confounded with differences in sample periods. Because overlapping windows induce serial dependence in the evaluation scores, uncertainty quantification in the next subsections combines standard DeLong approximations for ROC–AUC with time-aware calendar block-bootstrap procedures for PR–AUC and paired ROC–AUC differences, so as to better account for temporal dependence than purely independent and identically distributed resampling.

2.6 *Evaluation metrics and statistical inference*

We evaluate models along three axes: discrimination, decision quality under imbalance, and probability calibration. The primary discrimination metric is ROC–AUC, which measures the ability of a model to correctly rank spike and non-spike windows across all thresholds. Confidence intervals for ROC–AUC are obtained using the DeLong estimator, and paired model contrasts employ DeLong tests on the same instances [10], under the working approximation that evaluation windows are sufficiently weakly dependent for the standard independent and identically distributed-based variance formulas to provide a useful guide.

ROC–AUC can be insensitive to class imbalance and to the quality of predictions in high-risk regions. We therefore complement ROC–AUC with PR–AUC and threshold-based metrics (accuracy, precision, recall, and F1) evaluated at the pre-specified Youden-J and F_1 operating points. Confidence intervals for PR–AUC are constructed via a calendar block bootstrap with block length $B = 20$ trading days and $B^* = 500$ resamples, which respects temporal dependence and class clustering. Overall probabilistic accuracy and calibration are assessed via the Brier score and ECE formulated as

$$\text{Brier} = \frac{1}{n}\sum_{i=1}^{n}(\hat{p}_i - y_i)^2, \qquad \text{ECE} = \sum_{b=1}^{B_{\text{bin}}} \frac{n_b}{n}\,\big|\,\text{acc}(b) - \text{conf}(b)\big|.$$

Lower values of the Brier score indicate better average probabilistic accuracy, and lower ECE indicates better alignment between predicted probabilities and observed frequencies. We construct $B_{\text{bin}} = 10$ equal-frequency probability bins from the OOF training probabilities and keep their cutpoints fixed across models and on the test set. Calibration curves, Brier scores, and ECE are then computed on the test set using these shared bin boundaries, ensuring fair and comparable calibration diagnostics that are not tuned ex post to any single model. When multiple pairwise ROC–AUC comparisons are reported, we present the raw p-values from DeLong tests and from the calendar block bootstrap and interpret them descriptively, focusing on effect sizes and consistency across assets rather than on strict accept–reject decisions.

2.7 *Paired testing and bootstrap design*

All pairwise tests are paired on common evaluation dates so that model comparisons exploit the same underlying realizations and differences are not driven by sample composition. For ROC–AUC, paired DeLong tests directly account for the correlation between model scores on the same instances.

For PR–AUC, analytical variance formulas are less tractable under dependence, so we resort to the calendar block bootstrap described in Section 2.6. The test index is partitioned into consecutive, non-overlapping blocks; each bootstrap replicate resamples these blocks with replacement to form a pseudo-series on which PR–AUC is recomputed.

The same bootstrap resamples are used across models to induce common shocks and stabilize paired differences, yielding more reliable contrasts than independent resampling. This design aims to better accommodate dependence arising from both overlapping windows and volatility clustering than a bootstrap based on independent and identically distributed resampling.

2.8 *Assumptions and robustness safeguards*

The methodology rests on the following working assumptions and safeguards:

- Temporal causality —Model fitting, feature construction, and threshold selection use only past data; no future information enters at any stage.
- Dependence —Overlapping windows and volatility clustering induce serial correlation; inference combines standard DeLong approximations for ROC–AUC with block-based resampling for PR–AUC and paired ROC–AUC differences, in order to reduce sensitivity to strict independent and identically distributed assumptions.
- Alignment —Score vectors from different models are aligned by date to permit valid paired testing and to ensure that differences reflect model performance rather than sample variation.
- Label stability -The 70th-percentile spike threshold is fixed on the training split and not re-estimated on the test set, avoiding label drift and facilitating interpretation of prevalence shifts.
- Sensitivity —Results are examined across RF mtry values $\{\lfloor\sqrt{p}\rfloor, \lfloor p/3\rfloor, \lfloor p/2\rfloor\}$, with each value truncated below at one; the winning specification is consistent across CV folds and across assets, supporting the robustness of the chosen hyperparameters.

2.9 *End-to-end protocol*

For clarity and reproducibility, the end-to-end procedure from raw prices to final evaluation can be summarized as follows:

(i) Preprocess (train) —Winsorize returns at training quantiles $(q_{0.01}, q_{0.99})$; fit centering and scaling transformations; align the trading calendar across assets.

(ii) Label (train) —Construct windows X_i and define spike labels as $Y_i = \mathbf{1}_{\{\mathrm{RV}_{i\to i+5} > \mathrm{Quantile}_{0.70}(\mathrm{RV}_{\mathrm{train}})\}}$.

(iii) FDA (train) —Perform global principal component analysis on the window matrix; extract `fscore1`–`fscore4`; compute $\|\Delta X_i\|_2$ and $\|\Delta^2 X_i\|_2$; fit standardization parameters on the FDA feature block.

(iv) TDA (train) —Set τ to the first local minimum of the autocorrelation function of $|r_t|$; embed with $m = 3$; choose $\varepsilon_{\max} = 0.6$ times the median pairwise distance in the embedded cloud; run a Vietoris–Rips filtration with 40 radius levels; extract $\{T_{H_1}^{(1)}, T_{H_1}^{(2)}, T_{H_0}^{(1)}, E_{\mathrm{pers}}, B_1, P_{\mathrm{tot}}\}$.

(v) HAR (train) —Compute the HAR-type averages $h_i^{(d)}$, $h_i^{(w)}$, and $h_i^{(m)}$ from $|r_t|$ to encode short-, intermediate-, and longer-horizon lag-volatility information.

(vi) Cross-validation and tuning (train): perform rolling-origin five-fold cross-validation for each mtry $\in \{\lfloor\sqrt{p}\rfloor, \lfloor p/3\rfloor, \lfloor p/2\rfloor\}$, with each candidate truncated below at one. In each fold, fit RF models with $B = 500$ trees and `nodesize`$= 10$; select mtry by mean validation ROC–AUC and then refit the chosen specification on the full training split with $B = 1000$ trees.

(vii) Thresholds (train) —From OOF probabilities obtained on the training segment, select the Youden-J-optimal and F_1-optimal thresholds. Freeze these thresholds with all train-fitted objects (winsorization bounds, standardization parameters, embedding settings, filtration scales, and RF models).

(viii) Test (forward apply) -Apply all frozen transformations and thresholds to the test windows; score all models to obtain test-set probabilities; compute ROC–AUC (with DeLong confidence intervals), PR–AUC (with time-aware calendar block-bootstrap confidence intervals), and threshold-based classification metrics at the pre-specified F_1-optimal thresholds; summarize paired ROC–AUC contrasts using DeLong tests alongside time-aware calendar block-bootstrap ROC–AUC differences.

(ix) Calibration (test) -Using the fixed equal-frequency probability bins constructed from OOF training scores, compute Brier scores, expected calibration error (ECE), and reliability curves on the test set for all models.

3. Experimental setup

This section outlines the empirical environment in which the forecasting models are evaluated. We describe the assets, the construction of rolling return windows, the labeling and preprocessing rules, and the chronological train–test split that governs all stages of the analysis. The resulting study sample provides the foundation for the performance comparisons and calibration diagnostics reported in Section 4.

3.1 *Data and universe*

The empirical universe consists of four liquid United States equities: INTC, COP, NVDA, and DIS, observed on a daily frequency over 2010–2024. Adjusted closing prices are obtained from Yahoo Finance [34, 35, 36, 37]. Each asset is modeled independently: there is no cross-asset pooling or multi-task training, so all features, labels, and classifiers are estimated on an asset-by-asset basis. The sample for each asset is split in calendar order into an initial training segment and a subsequent test segment, with an approximate 70% to 30% split in terms of rolling windows. This design mirrors the temporal structure of a realistic deployment in which a model trained on historical data is applied to future observations, and it underpins the leakage controls described in Sections 2.2 and 2.5.

3.2 *Study sample and preprocessing*

Daily adjusted closing prices are first aligned to a unified trading calendar across all four assets and transformed into close-to-close log returns. Preprocessing follows the training-only protocol described in Section 2.2: winsorization, centering, and scaling parameters are estimated on the training segment and then applied forward in time to the test windows.

We construct overlapping return windows of length $W = 60$ days and define short-horizon volatility labels using the $K = 5$ day mean absolute return proxy from Section 2.2. Spike labels follow the five-day, 70th percentile rule defined in Section 2.2.

Table 2 summarizes the resulting study sample by asset, reporting raw observation counts, the number of rolling windows, the train–test split, and the prevalence of spike versus non-spike windows in each segment. Spike prevalence is locked at approximately 30% on the training segment by construction, whereas test-set prevalence varies substantially across assets (approximately 65% for INTC, 50% for COP, 57% for NVDA, and 52% for DIS).

This variation motivates the joint use of area under the ROC–AUC, area under the precision–recall curve (PR–AUC), threshold-based metrics, and calibration diagnostics in Section 4. Table 1 reports the corresponding delay-embedding settings (embedding dimension $m = 3$, data-driven delay τ), training-only winsorization bounds, and the realized spike thresholds for each asset.

Table 2.: Study sample summary by asset

Asset	Raw observed	Window count	Train segment N_{train}	Spike train	Normal train	Spike ratio train	Test segment N_{test}	Spike test	Normal test	Spike ratio test
INTC	3771	3707	2594	778	1816	0.300	1113	718	395	0.645
COP	3771	3707	2594	778	1816	0.300	1113	562	551	0.505
NVDA	3771	3707	2594	778	1816	0.300	1113	635	478	0.571
DIS	3771	3707	2594	778	1816	0.300	1113	582	531	0.523

The identical raw observation counts and window totals across assets reflect the unified calendar alignment. Zero missing prices imply that no imputation is required prior to window construction. The deliberate choice of a fixed training spike ratio (approximately 30%) combined with asset-specific test ratios underpins the emphasis on PR–AUC, threshold metrics, and calibration summaries in Tables 3 and 5 and in Figure 2, where performance is evaluated under realistically nonstationary class balances.

3.3 *Cross-validation and hyperparameters*

The cross-validation protocol and hyperparameter selection are designed to mirror a rolling deployment and to keep model choice strictly pre-specified. We use rolling-origin five-fold cross-validation for each asset–feature-family combination (FDA, TDA, and the combined specification BOTH), respecting calendar order in every fold and prohibiting any shuffling of observations. In each fold, a contiguous block of early windows serves as training data and a subsequent block serves as validation data, so that validation occurs strictly after training, consistent with the time-aware principles described in Sections 2.5 and 2.6.

RF classifiers employ the pre-specified tuning grid introduced in Section 2.4 for the number of variables sampled at each split: $mtry \in \{\lfloor\sqrt{p}\rfloor, \lfloor p/3\rfloor, \lfloor p/2\rfloor\}$, with each candidate truncated below at one, where p is the dimensionality of the feature set under consideration. All remaining hyperparameters are held fixed across assets and feature families: we use $B = 500$ trees and a minimum node size of 10 during cross-validation, and $B = 1000$ trees with the same node size when refitting models on the full training segment.

For each candidate $mtry$ value and model family, we fit forests on the training portion of each fold and evaluate ROC–AUC on the corresponding validation portion. The $mtry$ value that maximizes the mean cross-validated ROC–AUC across the five folds is then selected and held fixed when refitting on the full training segment and when constructing out-of-fold probabilities for threshold selection.

Table 6 in Section 4.5 reports the resulting mean and standard deviation of cross-validated ROC–AUC for each asset and feature family, with the selected $mtry$ values. The close alignment between the cross-validation ranking in Table 6 and the test-set performance in Tables 3 and 4 confirms the suitability of this rolling-origin cross-validation design as a stable basis for pre-specifying hyperparameters within the end-to-end protocol summarized in Section 2.9.

4. Results

This section reports the main empirical findings of the forecasting study. We first compare the discrimination and decision performance of the competing model families across all assets, and then analyze paired differences, calibration properties, and cross-validation behavior. The last subsection summarizes patterns in the feature sets and their relation to spike and non-spike episodes.

4.1 *Main discrimination and decision metrics*

Table 3 reports the out-of-sample discrimination and decision metrics for all four assets and model families, while Figure 1 displays the corresponding ROC curves. Table 3 and Figure 1 document the main pattern that the integrated specification BOTH consistently improves on the FDA-only and TDA-only baselines across all four assets. In terms of ROC–AUC, the BOTH specification achieves the highest test performance among the RF families (FDA, TDA, BOTH) for each asset. Across INTC, COP, NVDA, and DIS, its ROC–AUC values lie in the 0.65 to 0.77 range and improve on the corresponding FDA and TDA baselines by roughly 3 to 8 percentage points (Table 3). The associated ROC envelopes in Figure 1 show visibly higher curves for BOTH over a broad range of false-positive rates, indicating a consistent gain in ranking quality rather than an isolated improvement at a particular operating point.

Relative to the HAR-only XGBoost benchmark, the picture is more nuanced: BOTH dominates on COP, is slightly ahead on NVDA, and is outperformed by XGBoost on INTC and DIS. PR–AUC and F1 follow a similar pattern, with BOTH and XGBoost generally delivering the strongest decision-oriented performance under the pronounced class imbalance and asset-specific spike prevalences reported in Table 2.

At the pre-specified F_1 operating point, INTC and COP are operated in a deliberately conservative regime with recall close to one and moderate precision, reflecting a preference for catching most spike episodes. NVDA exhibits a more balanced precision–recall trade-off and attains the highest F1 under BOTH, while DIS shows that BOTH and XGBoost can achieve similar ROC–AUC and PR–AUC yet differ meaningfully once calibration is taken into account (Section 4.3). Overall, Table 3 and Figure 1 jointly indicate that combining FDA, TDA, and HAR features yields systematically better classification performance than either block in isolation and is competitive with the HAR-only XGBoost benchmark, being clearly ahead on COP, roughly on par on NVDA, and somewhat behind on INTC and DIS.

Table 3.: Test performance: FDA, TDA, BOTH, XGBoost

Asset	Model	N_{test}	AUC	CI		PR-AUC	PR-CI		Threshold	Accuracy	Precision	Recall	F1
				low	high		low	high					
INTC	FDA	1113	0.6495	0.6157	0.6834	0.7609	0.6975	0.8215	0.186	0.6550	0.6564	0.9763	0.7850
INTC	TDA	1113	0.6375	0.6040	0.6709	0.7551	0.6859	0.8173	0.000	0.6451	0.6451	1.0000	0.7843
INTC	BOTH	1113	0.6712	0.6375	0.7049	0.7537	0.6733	0.8271	0.260	0.6873	0.6923	0.9276	0.7929
INTC	XGB	1113	0.7121	0.6803	0.7439	0.7965	0.7155	0.8482	0.292	0.6981	0.7045	0.9164	0.7966
COP	FDA	1113	0.7373	0.7083	0.7664	0.6958	0.6257	0.7831	0.194	0.6244	0.5788	0.9413	0.7168
COP	TDA	1113	0.6884	0.6576	0.7192	0.6911	0.6027	0.7758	0.240	0.6020	0.5696	0.8665	0.6874
COP	BOTH	1113	0.7662	0.7386	0.7939	0.7361	0.6570	0.8094	0.346	0.6999	0.6492	0.8826	0.7481
COP	XGB	1113	0.7439	0.7154	0.7725	0.7288	0.6419	0.8112	0.263	0.6667	0.6169	0.8968	0.7310
NVDA	FDA	1113	0.6764	0.6449	0.7080	0.7285	0.6400	0.8081	0.382	0.6451	0.6695	0.7465	0.7059
NVDA	TDA	1113	0.6365	0.6041	0.6689	0.7092	0.5999	0.8001	0.146	0.5750	0.5760	0.9669	0.7219
NVDA	BOTH	1113	0.6907	0.6596	0.7219	0.7352	0.6327	0.8311	0.332	0.6550	0.6551	0.8346	0.7341
NVDA	XGB	1113	0.6891	0.6577	0.7204	0.7360	0.6218	0.8283	0.314	0.6406	0.6474	0.8126	0.7207
DIS	FDA	1113	0.6195	0.5867	0.6523	0.6485	0.5442	0.7482	0.190	0.5391	0.5361	0.8814	0.6667
DIS	TDA	1113	0.5898	0.5566	0.6230	0.6488	0.5478	0.7451	0.076	0.5175	0.5206	0.9759	0.6790
DIS	BOTH	1113	0.6666	0.6350	0.6982	0.7067	0.6044	0.7810	0.074	0.5238	0.5235	0.9966	0.6864
DIS	XGB	1113	0.6823	0.6511	0.7135	0.7073	0.6178	0.7868	0.509	0.6361	0.7411	0.4674	0.5732

4.2 *Pairwise AUC comparisons*

While Table 3 and Figure 1 summarize absolute asset–model performance, Table 4 reports paired ROC–AUC contrasts. For each asset–comparison pair, it lists DeLong p-values with time-aware block-bootstrap ROC–AUC differences, $\widehat{\Delta}_{\text{BB}}$, and their percentile confidence intervals and two-sided bootstrap p-values. We report raw p-values and interpret them descriptively, focusing on effect sizes and consistency across assets rather than on strict accept–reject decisions.

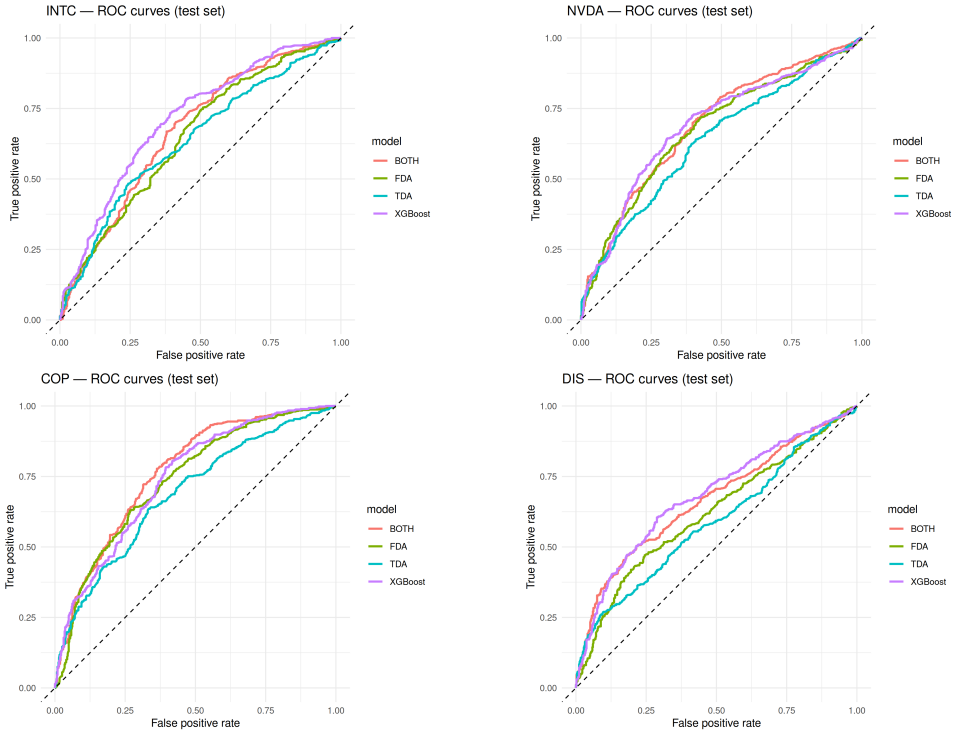

Figure 1.: Test ROC curves by asset for FDA, TDA, BOTH, and XGBoost. Panels (top left to bottom right): INTC, NVDA, COP, DIS.

The DeLong contrasts in Table 4 generally favor BOTH over TDA and FDA, but the strength of the evidence varies by asset. For COP, NVDA, and DIS, TDA versus BOTH yields very small p-values and positive block-bootstrap differences, indicating clear improvements in ROC–AUC when TDA features are combined with FDA and HAR. On INTC, the TDA versus BOTH comparison is directionally favorable to BOTH but only marginally significant, and the FDA versus BOTH contrast is not statistically decisive. Comparisons against XGBoost are mixed: for COP the integrated specification BOTH exhibits a positive ROC–AUC difference over XGBoost, whereas for INTC and DIS XGBoost attains somewhat higher ROC–AUC, and for NVDA the two models are essentially indistinguishable.

The block-bootstrap columns of Table 4 provide a more conservative perspective that explicitly acknowledges serial dependence. The point estimates $\widehat{\Delta}_{\mathrm{BB}}$ are uniformly positive when contrasting BOTH with TDA and FDA, but the corresponding confidence intervals occasionally include zero, especially for INTC and COP, reflecting wider uncertainty under dependence and overlapping windows. For COP, NVDA, and DIS, the bootstrap intervals for TDA versus BOTH are strictly positive, reinforcing the conclusion that integrating functional and topological features yields genuine gains in ranking performance relative to TDA alone. By contrast, the bootstrap comparisons between XGBoost and BOTH highlight that the HAR-only benchmark remains very competitive in some cases, rather than being uniformly dominated.

4.3 *Probability Calibration*

Table 5 collects Brier scores and expected calibration errors (ECE) for all asset–model combinations, whereas Figure 2 presents the corresponding reliability curves. These diagnostics probe a different aspect of performance than ROC and precision–recall metrics: the alignment between predicted probabilities and empirical event frequencies, with the Brier score reflecting overall probabilistic accuracy (calibration plus sharpness) and ECE focusing specifically on calibration.

Table 4.: Paired ROC–AUC comparisons (DeLong and block bootstrap)

Asset	Comparison	DeLong p-value	BB diff	BB CI (low)	BB CI (high)	BB p-value
INTC	FDA versus TDA	0.5642	-0.0079	-0.0779	0.0560	0.780
INTC	TDA versus BOTH	0.0517	0.0280	-0.0332	0.0879	0.356
INTC	FDA versus BOTH	0.2515	0.0204	-0.0435	0.0865	0.532
INTC	XGB versus BOTH	3.7×10^{-5}	-0.0377	-0.0790	-0.0012	0.048
COP	FDA versus TDA	0.0010	-0.0458	-0.1056	0.0166	0.136
COP	TDA versus BOTH	2.9×10^{-11}	0.0739	0.0260	0.1243	0.004
COP	FDA versus BOTH	0.0099	0.0277	-0.0112	0.0686	0.228
COP	XGB versus BOTH	0.0198	0.0235	-0.0104	0.0559	0.116
NVDA	FDA versus TDA	0.0181	-0.0469	-0.0959	0.0088	0.088
NVDA	TDA versus BOTH	8.8×10^{-5}	0.0539	0.0072	0.1006	0.024
NVDA	FDA versus BOTH	0.2822	0.0122	-0.0295	0.0552	0.612
NVDA	XGB versus BOTH	0.8660	0.0026	-0.0266	0.0344	0.892
DIS	FDA versus TDA	0.1073	-0.0305	-0.0936	0.0294	0.360
DIS	TDA versus BOTH	9.3×10^{-6}	0.0726	0.0067	0.1455	0.020
DIS	FDA versus BOTH	6.5×10^{-4}	0.0443	-0.00022	0.0900	0.056
DIS	XGB versus BOTH	0.1562	-0.0161	-0.0548	0.0263	0.468

Across assets, BOTH systematically improves Brier scores and ECE relative to the FDA-only and TDA-only baselines, indicating that the richer feature set not only sharpens ranking but also yields better-behaved probability forecasts. Compared with the HAR-only XGBoost benchmark, the picture is asset-dependent: for COP, BOTH attains lower Brier score and ECE; for INTC, XGBoost achieves the lowest Brier score and ECE; and for NVDA and DIS the two models are broadly comparable, with small trade-offs between Brier score and ECE (for example, BOTH slightly better in Brier score for NVDA and in ECE for DIS, and XGBoost holding the opposite advantage). Overall, these patterns suggest that combining FDA and TDA with HAR lags improves calibration over single-family feature sets and delivers calibration quality that is broadly in line with, and in some cases close to or better than, a strong HAR-only tree-based benchmark.

Figure 2 illustrates how these differences materialize across the probability range. The reliability curves for BOTH and XGBoost typically lie closest to the 45° line, especially in the mid-probability region where most predictions are concentrated. In contrast, FDA and TDA alone tend to display underconfidence at low predicted probabilities and overconfidence near the upper deciles, with more pronounced deviations from perfect calibration. Table 5 and Figure 2 show that combining functional and topological features not only improves the ranking of spike episodes but also leads to better-calibrated risk scores than single-family baselines, an important property for downstream decision rules that hinge on probability thresholds.

Table 5.: Calibration metrics (Brier score and ECE)

Asset	Model	Brier	ECE
INTC	FDA	0.2522	0.1946
INTC	TDA	0.2739	0.2314
INTC	BOTH	0.2390	0.1639
INTC	XGBoost	0.2193	0.1429
COP	FDA	0.2177	0.0833
COP	TDA	0.2337	0.1124
COP	BOTH	0.1987	0.0474
COP	XGBoost	0.2090	0.0719
NVDA	FDA	0.2361	0.1008
NVDA	TDA	0.2501	0.1137
NVDA	BOTH	0.2274	0.0855
NVDA	XGBoost	0.2284	0.0809

Table 5 continues in the next page

Asset	Model	Brier	ECE
DIS	FDA	0.2624	0.1414
DIS	TDA	0.2717	0.1598
DIS	BOTH	0.2418	0.1062
DIS	XGBoost	0.2404	0.1252

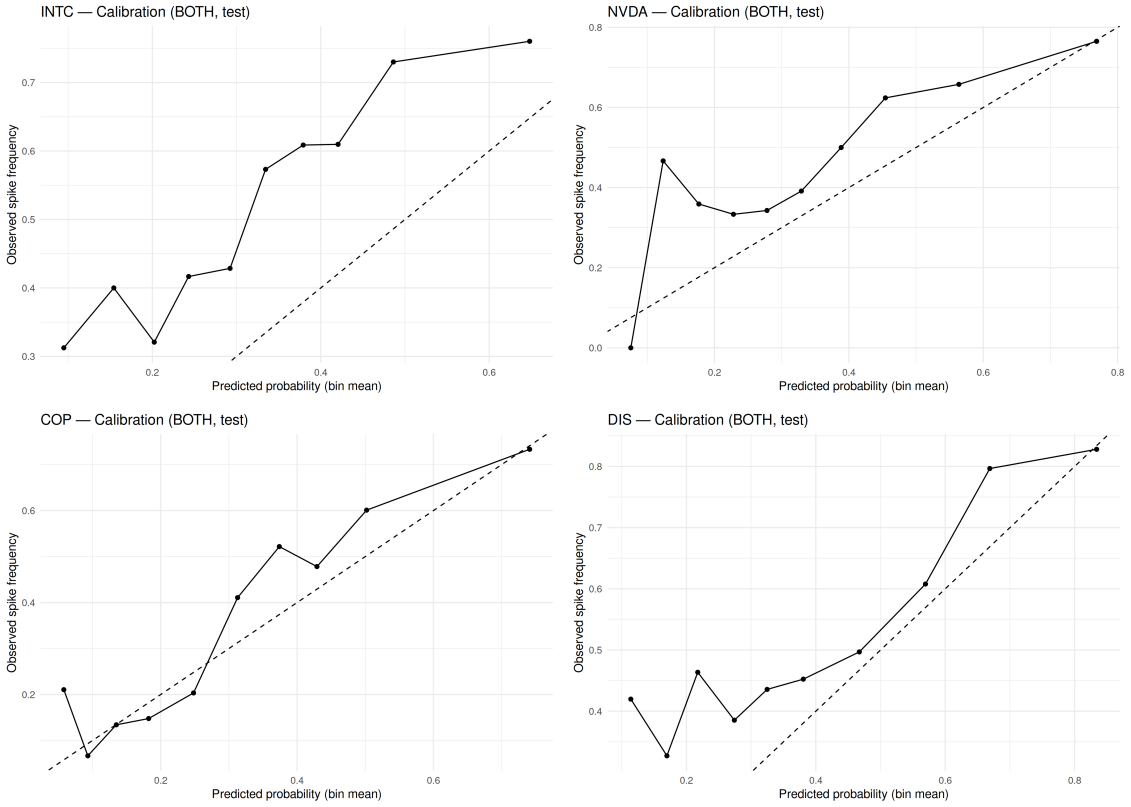

Figure 2.: Reliability curves and calibration diagnostics by asset. Panels (top left to bottom right): INTC, NVDA, COP, DIS.

4.4 *Feature behavior and class separation*

To shed light on how the features behave across spike and non-spike windows, Figure 3 displays representative TDA and FDA summaries stratified by class for each asset. The panels include total persistence, Betti-1 summaries, leading FPCA scores, and discrete derivative norms, thereby linking the quantitative performance gains in Tables 3–5 to concrete feature-level patterns.

A robust qualitative pattern emerges across assets: spike windows tend to exhibit higher topological complexity in the delay-embedded state space, as reflected in elevated total persistence and Betti-1 statistics, while the functional scores and derivative norms capture complementary changes in level, slope, and roughness of recent return paths.

For instance, NVDA and COP show clear upward shifts in both topological intensity and roughness-based FDA measures for spike windows, consistent with their relatively large ROC–AUC and PR–AUC improvements under BOTH. DIS displays more modest separation at the level of individual summaries, which helps explain why its performance gains are more muted, even though the combined specification still improves calibration and overall discrimination.

By design, Figure 3 is descriptive rather than inferential. Nonetheless, the systematic shifts in both topological and functional features between classes provide a structural interpretation of the superiority of BOTH: topology reacts to regime geometry and clustering in the reconstructed state space, whereas functional summaries encode smoother, low-frequency aspects of the path. Their combination furnishes a richer representation of pre-spike dynamics than either block alone.

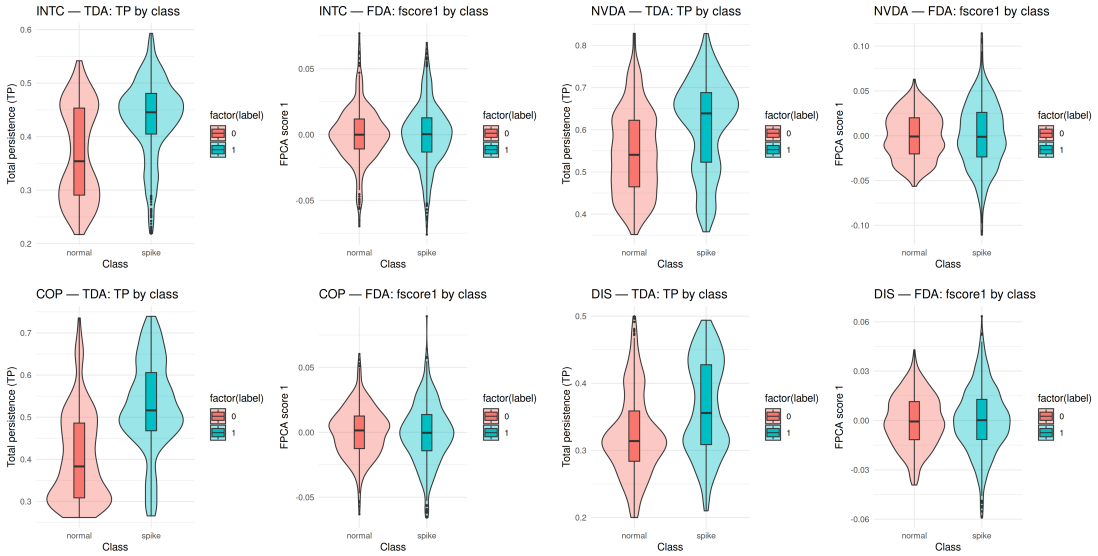

Figure 3.: Selected TDA and FDA feature summaries by class (spike versus normal). Panels (top left to bottom right): INTC, NVDA, COP, DIS.

4.5 *Cross-validation and tuning summary*

Finally, Table 6 summarizes the rolling-origin cross-validation results used for model selection, reporting mean ROC–AUC and its standard deviation across folds for each asset and feature family, with the chosen *mtry* values. This table links the pre-specified tuning procedure from Section 2.4 to the out-of-sample performance documented above.

The cross-validation results in Table 6 show that the relative ordering of feature families is broadly consistent with the test-set ROC–AUCs, but not uniformly dominated by any single specification. For INTC and NVDA, BOTH attains the highest mean cross-validated ROC–AUC, mirroring its advantage over FDA and TDA on the test set. For COP and DIS, the best-performing single-family baseline (FDA for COP, TDA for DIS) is slightly ahead of BOTH in mean cross-validated ROC–AUC, although the differences are modest and fall within one standard deviation. The selected *mtry* values for BOTH lie in the lower end of the grid (around three variables per split), and forests are consistently trained with $B = 500$ trees during cross-validation and $B = 1000$ trees in the final fits, supporting the use of this rolling-origin cross-validation scheme as a stable basis for pre-specifying hyperparameters within the overall end-to-end protocol.

In addition to the RF families in Table 6, we include a gradient boosting tree benchmark (XGBoost; [38]) fitted on the HAR block only. Its hyperparameters (tree depth, learning rate, number of trees, and subsampling rates) are fixed ex ante and kept constant across assets under the same chronological constraints, so that XGBoost provides a strong but transparent nonlinear baseline built solely from lagged-volatility cues. The richer FDA/TDA/HAR representations in Tables 3 and 5 can then be interpreted relative to this HAR-only benchmark.

Tables 3, 4, 5, and 6, along with Figures 1, 2, and 3, show that the FDA/TDA/HAR specification systematically improves discrimination, calibration, and interpretability over single-family feature sets and is competitive with a HAR-only gradient-boosted tree benchmark, under a validation scheme that respects the temporal structure of the data.

Table 6.: RF cross-validation summary

Asset	Model	mtry	Mean CV AUC	SD CV AUC
INTC	FDA	2	0.5698	0.0927
INTC	TDA	3	0.5180	0.0891
INTC	BOTH	3	0.5865	0.0540
COP	FDA	2	0.6310	0.0476
COP	TDA	2	0.6207	0.1136
COP	BOTH	3	0.6128	0.1801
NVDA	FDA	2	0.5657	0.0578
NVDA	TDA	2	0.5502	0.1253
NVDA	BOTH	3	0.5876	0.1029
DIS	FDA	2	0.5374	0.1274
DIS	TDA	2	0.5395	0.0688
DIS	BOTH	3	0.5331	0.1500

5. Simulation study

Our simulation study examines the behavior of the proposed feature sets under a controlled, regime-switching volatility environment. Conditional returns are generated from a two-state regime-switching generalized autoregressive conditional heteroskedasticity (Regime-GARCH) process in which a latent Markov chain switches between low and high volatility regimes that differ in both unconditional variance and persistence. Innovations follow a heavy-tailed Student-t distribution to induce realistic spike-and-cluster patterns, in line with recent work on tail-adaptive random number generation for heavy-tailed models [39]. Labeling, feature construction, and the learning method follow exactly the chronological protocol in Section 2; only the data-generating mechanism changes. This keeps the simulation strictly comparable to the empirical study.

5.1 *Discrimination and decision metrics*

Table 7 summarizes test-set discrimination and decision metrics under the Regime-GARCH scenario, while Figure 4 displays the corresponding ROC curves. In this controlled environment, all models achieve moderate ROC–AUC, with values ranging from about 0.49 (TDA) to 0.62 (XGBoost). The HAR-only XGBoost benchmark attains the highest ROC–AUC (approximately 0.62) and PR–AUC (about 0.32), followed by the combined specification BOTH (ROC–AUC around 0.57, PR–AUC around 0.26), with FDA and TDA slightly behind. Thus, when the data-generating mechanism is close to a low-dimensional Regime-GARCH and HAR structure, a parsimonious lag-volatility model can dominate richer FDA- and TDA-based feature sets in terms of pure discrimination.

The decision-oriented metrics in Table 7 convey a similar message. At the pre-specified F_1 operating point, accuracies lie between roughly 0.56 and 0.69 and F1 scores between about 0.25 and 0.38, with XGBoost typically delivering the best trade-off between precision and recall and BOTH improving on FDA and TDA. Table 7 and Figure 4 show that, under a parametric Regime-GARCH process, lagged-volatility information remains highly informative and can outperform more flexible FDA- and TDA-based specifications.

Table 7.: Simulation: test performance under the Regime-GARCH scenario

Scenario	Model	N_{test}	AUC	CI low	CI high	PR-AUC	PR-CI low	PR-CI high	Threshold	Accuracy	Precision	Recall	F1
Regime GARCH	FDA	1781	0.5425	0.5090	0.5760	0.2416	0.1893	0.3045	0.3080	0.6721	0.2339	0.2715	0.2513
Regime GARCH	TDA	1781	0.4924	0.4593	0.5255	0.1991	0.1484	0.2662	0.2680	0.5592	0.1980	0.3850	0.2615
Regime GARCH	BOTH	1781	0.5694	0.5345	0.6043	0.2608	0.1877	0.3482	0.3000	0.6126	0.2504	0.4571	0.3235
Regime GARCH	XGBoost	1781	0.6232	0.5892	0.6572	0.3162	0.2380	0.4038	0.2888	0.6884	0.3197	0.4765	0.3826

Figure 4.: ROC curves for FDA, TDA, BOTH, and XGBoost under the regime-GARCH scenario.

5.2 *Pairwise AUC contrasts*

Whereas Table 7 and Figure 4 document absolute performance levels, Table 8 focuses on paired ROC–AUC differences under the same simulated paths. The DeLong p-values quantify whether the observed ROC–AUC gaps are statistically detectable when predictions are compared on identical test instances, and the block-bootstrap columns provide a complementary assessment that accounts for temporal dependence.

Table 8 shows that augmenting topological summaries with functional and HAR features yields a detectable improvement over TDA alone: in the TDA versus BOTH row, the ROC–AUC difference $\hat{\Delta}_{\text{BB}} = \text{AUC(BOTH)} - \text{AUC(TDA)}$ is positive (around 0.08), with a bootstrap interval entirely above zero and a very small DeLong p-value. The FDA versus BOTH contrast is directionally favorable to BOTH but not statistically decisive, reflecting a smaller ROC–AUC gap and a bootstrap interval that crosses zero.

By contrast, the XGBoost versus BOTH comparison features a negative block-bootstrap difference (the same definition $\text{AUC(BOTH)} - \text{AUC(XGBoost)}$), with a confidence interval that lies largely below zero and a very small p-value, indicating that in this simulation the HAR-only gradient-boosted tree benchmark achieves a higher ROC–AUC than the integrated specification. Together with Table 7, these contrasts underline that the empirical ranking in favor of BOTH is not universal and can reverse when the data-generating process is particularly well aligned with lagged-volatility predictors.

Table 8.: Simulation: paired AUC comparisons under the Regime-GARCH scenario (BB diff = $\widehat{\Delta}_{\mathrm{BB}}$).

Scenario	Comparison	DeLong p	BB diff	BB CI (low)	BB CI (high)	BB p
Regime-GARCH	FDA versus TDA	0.017	-0.048	-0.119	0.017	0.144
Regime-GARCH	TDA versus BOTH	0.0001	0.077	0.008	0.144	0.020
Regime-GARCH	FDA versus BOTH	0.145	0.031	-0.024	0.097	0.316
Regime-GARCH	XGB versus BOTH	0.0000	-0.047	-0.094	-0.001	0.048

5.3 *Calibration diagnostics*

Calibration properties in simulation are summarized in Table 9, with the corresponding reliability diagrams shown in Figure 5. As in the empirical analysis, these diagnostics evaluate how closely predicted probabilities align with realized spike frequencies, complementing the discrimination-oriented ROC–AUC and PR–AUC metrics.

In contrast to the empirical setting, XGBoost attains the lowest Brier score and the lowest expected calibration error in the Regime-GARCH scenario. This reflects both its strong discrimination and its well-calibrated probabilities when the underlying volatility dynamics are close to a low-dimensional lag-volatility process. The combined specification BOTH improves calibration relative to TDA and is broadly comparable to FDA, but it does not outperform the HAR-only XGBoost benchmark in this particular design. Viewed jointly, Tables 7–9 and Figures 4–5 indicate that, under a correctly specified Regime-GARCH and HAR structure, a parsimonious lag-volatility model can dominate more flexible FDA- and TDA-based specifications, whereas in the empirical setting the richer feature set yields clearer improvements over single-family baselines.

Table 9.: Simulation: calibration metrics under the Regime-GARCH scenario

Scenario	Model	Brier	ECE
Regime-GARCH	FDA	0.172	0.086
Regime-GARCH	TDA	0.193	0.133
Regime-GARCH	BOTH	0.174	0.103
Regime-GARCH	XGBoost	0.159	0.058

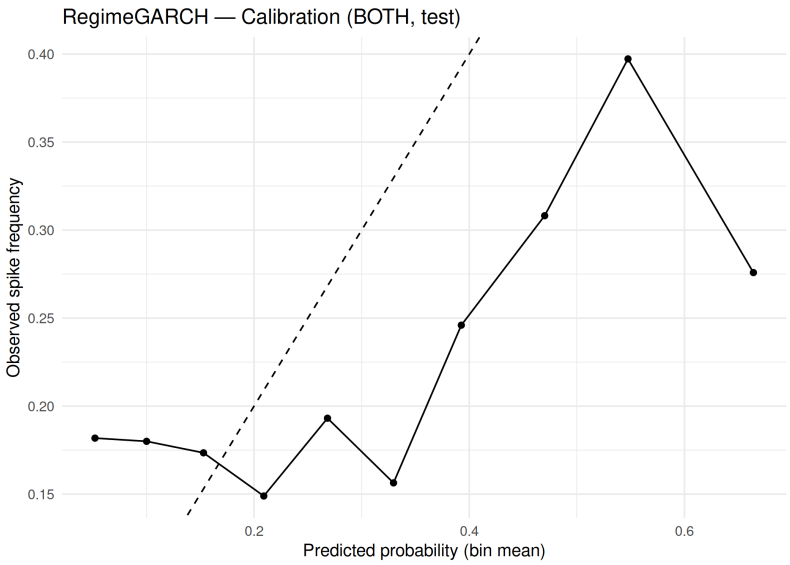

Figure 5.: Reliability curves and calibration diagnostics under the regime-GARCH scenario.

6. Conclusions

This article investigates whether combining functional and topological summaries of recent return paths improves the prediction of short-horizon volatility spikes. Using a train-only, rolling-origin learning protocol with paired inference, we compare functional features, topological features, and the combined FDA, TDA, and HAR specification against an HAR-only XGBoost benchmark. Across four large-cap equities, the pre-specified combined specification delivers stronger discrimination and decision performance than the single-family FDA and TDA models, and it improves probabilistic accuracy and calibration. Relative to the HAR-only benchmark, the combined specification is clearly ahead on some assets, roughly comparable on others, and somewhat behind on a few cases, indicating that the relative performance of nonlinear approaches is asset-dependent.

Paired DeLong tests and time-aware block-bootstrap contrasts quantify these differences, while Brier scores and expected calibration error show that the combined specification consistently improves calibration relative to the single-family baselines and is broadly competitive with the HAR-only model.

The asset-level results exhibit a coherent pattern. COP and DIS show the largest and most stable improvements under the combined specification, with clear separation in both ROC curves and class-conditional feature summaries. INTC and NVDA also benefit from the combined feature set, although block-bootstrap intervals around some AUC differences include zero, reflecting serial dependence and asset-specific prevalence shifts. DIS provides an informative case where the HAR-only benchmark attains slightly higher AUC and PR-AUC, yet the combined specification yields better expected calibration error and comparable Brier scores, illustrating that probability calibration should be assessed alongside ranking metrics when models are used for risk-based decisions.

From an operational perspective, three recommendations follow. First, strict train–calibrate–test chronology and the associated leakage controls should be treated as design principles: all transformations, feature mappings, and thresholds should be fitted on the training split and applied forward in time, with continuous monitoring of class prevalence, AUC, PR-AUC, and expected calibration error to detect drift. Second, threshold policies should be viewed as adjustable levers on top of a fixed score. Institutions with different precision–recall requirements can reoptimize thresholds on the same calibration split without retraining the underlying models.

Third, model risk management frameworks should monitor a small but complementary set of metrics—AUC, PR-AUC, Brier score, and expected calibration error—and establish guardrails for acceptable deviations from the benchmark values reported in this study. The end-to-end protocol in Section 2.9 provides a concrete blueprint for such monitoring.

The analysis also has limitations that define the scope of the conclusions. The empirical study focuses on four large-cap equities and a single spike definition based on a fixed training quantile and a five-day horizon. Different markets, alternative horizons, or alternative labeling rules could alter both spike prevalence and classification difficulty. Dependence is addressed through paired DeLong tests and block-based resampling rather than independent and identically distributed assumptions, but volatility clustering and structural breaks can still widen bootstrap intervals. The feature engineering strategy centers on a specific and interpretable set of FDA and TDA summaries; alternative learners or sequence-based feature extractors might yield further improvements at the cost of transparency. Finally, the Regime-GARCH simulation shows that under a volatility process that closely follows a low-dimensional lag-volatility structure, a parsimonious benchmark can outperform more flexible specifications in both discrimination and calibration.

Several extensions follow naturally. Methodologically, richer topological descriptors such as signed Betti curves or persistence landscapes, as well as interaction terms between functional and topological features, may enhance separation between spike and non-spike windows. On the calibration side, drift-aware techniques such as time-varying thresholds, adaptive recalibration schedules, or distribution-free conformal adjustments offer promising avenues for maintaining reliability under shifting market conditions. Cross-sectionally, extending the framework to a broader asset universe or to multi-asset models that exploit joint information could improve generality and support portfolio-level spike management. Integrating exogenous information such as scheduled macroeconomic releases, news-based indicators, or option-implied measures may also help distinguish endogenous path geometry from event-driven shocks.

Overall, the evidence supports the use of the combined FDA, TDA, and HAR specification as a practical default for short-horizon spike prediction. It improves discrimination, enhances probability calibration, and provides interpretable feature-level signals, all within a validation scheme that respects the temporal structure of financial data. With appropriate recalibration and monitoring, the proposed framework can serve as a robust component of volatility-aware risk management and forecasting systems.

Appendix: R code for the volatility spike forecasting method (FDA + TDA)

```
## ============================================================================================
## Short-Horizon Volatility Spike Forecasting FDA + TDA + HAR method + Regime-GARCH simulation
## Fidelity to article; uses up to 32 cores via doParallel
## Output: outfTDA2/tables/*.csv, outfTDA2/plots/*.png
## ============================================================================================

set.seed(123)
## ---------- Packages ----------
pkgs <- c("quantmod", "TDA", "randomForest", "zoo", "dplyr",  "ggplot2", "pROC", "PRROC", "data.table",
"purrr", "stringr", "cowplot", "gridExtra", "moments",  "xgboost", "parallel", "doParallel", "foreach")
miss <- setdiff(pkgs, rownames(installed.packages()))
if (length(miss)) install.packages(miss, dependencies = TRUE)
invisible(lapply(pkgs, library, character.only = TRUE))
options(dplyr.summarise.inform = FALSE)
## ---------- Parallel setup (up to 32 cores) ----------
n_detected <- parallel::detectCores()
if (is.na(n_detected)) n_detected <- 1L
N_CORES <- min(32L, n_detected)
cl <- parallel::makeCluster(N_CORES)
doParallel::registerDoParallel(cl)
on.exit({
  try(parallel::stopCluster(cl), silent = TRUE)
  doParallel::registerDoSEQ()
}, add = TRUE)
## ---------- Global parameters ----------
ASSETS     <- c("INTC", "COP", "NVDA", "DIS")
FROM       <- "2010-01-01"
TO         <- "2024-12-30"
WINDOW_LEN <- 60L            # past window length
K_RV       <- 5L             # forward horizon for volatility proxy
SPIKE_Q    <- 0.70           # spike threshold quantile (train-only)
TRAIN_FRAC <- 0.70           # train fraction at window level
NHARM_FDA  <- 4L             # number of FPCA components (scores)
BLOCK_LEN  <- 20L            # block length for bootstrap in days
N_BOOT     <- 500L           # number of bootstrap replicates
B_BIN      <- 10L            # number of bins for calibration (ECE)
OUTDIR  <- "outfTDA2"
TABDIR  <- file.path(OUTDIR, "tables")
PLOTDIR <- file.path(OUTDIR, "plots")
dir.create(OUTDIR,  showWarnings = FALSE, recursive = TRUE)
dir.create(TABDIR,  showWarnings = FALSE, recursive = TRUE)
dir.create(PLOTDIR, showWarnings = FALSE, recursive = TRUE)
## ============================================================
## 1. Helper functions
```

```
## ============================================================
## ---- Data fetch from Yahoo ----
get_prices <- function(sym, from = FROM, to = TO) {
  message("=== Fetching: ", sym, " ===")
  X <- tryCatch(quantmod::getSymbols(sym, src = "yahoo", from = from, to = to, auto.assign = FALSE),
    error = function(e) NULL)
  if (is.null(X)) {
    warning("[WARN] No data for ", sym)
    return(NULL)
  }
  cp_raw   <- quantmod::Cl(X)
  n_na_raw <- sum(is.na(cp_raw))
  cp       <- cp_raw
  if (any(is.na(cp))) {
    cp <- na.locf(cp)
    cp <- na.locf(cp, fromLast = TRUE)
  }
  return(list(
    close = cp, n_na_raw = n_na_raw
  ))
}
## ---- Embedding parameter selection (m=3, tau via first local ACF minimum) ----
select_embedding_params <- function(r_abs, max_lag = 20L) {
  r_abs <- r_abs[is.finite(r_abs)]
  if (length(r_abs) < 10L) {
    return(list(m = 3L, tau = 1L))
  }
  ac_full <- acf(r_abs, lag.max = max_lag, plot = FALSE, na.action = na.pass)$acf
  ac <- ac_full[-1]  # drop lag 0
  tau <- NA_integer_
  if (length(ac) >= 3L) {
    for (k in 2:(length(ac) - 1L)) {
      if (is.finite(ac[k - 1]) && is.finite(ac[k]) && is.finite(ac[k + 1]) &&
          ac[k] < ac[k - 1] && ac[k] <= ac[k + 1]) {
        tau <- k
        break
      }
    }
  }
  if (is.na(tau) || tau < 1L) {
    tau <- which.min(ac)
    if (is.na(tau) || tau < 1L) tau <- 1L
  }
  return(list(m = 3L, tau = tau))
}
## ---- Time-delay embedding ----
embed_curve <- function(y, dim = 3L, tau = 1L) {
  n    <- length(y)
  mlen <- n - (dim - 1L) * tau
  if (mlen <= 1L) return(NULL)
  mat <- matrix(NA_real_, nrow = mlen, ncol = dim)
  for (j in seq_len(dim)) {
    mat[, j] <- y[(1L + (j - 1L) * tau):(mlen + (j - 1L) * tau)]
  }
  return(mat)
}
auto_maxscale <- function(emb, factor = 0.6) {
  d <- as.numeric(dist(emb))
  if (length(d) == 0L) return(0.5)
  return(factor * median(d, na.rm = TRUE))
}
## ---- TDA feature extractor (on |r_t|, robust) ----
tda_features_safe <- function(y_window, dim_embed = 3L, tau = 1L, maxdim = 1L) {
  out0 <- c(top1_H1 = 0, top2_H1 = 0, top1_H0 = 0, pentropy = 0, betti1 = 0, total_pers = 0)
  emb <- embed_curve(y_window, dim = dim_embed, tau = tau)
  if (is.null(emb) || nrow(emb) < 5L) return(out0)
  maxscale <- tryCatch(auto_maxscale(emb), error = function(e) 0.5)
  pd <- tryCatch(
    ripsDiag(emb, maxdimension  = maxdim, maxscale = maxscale,
             dist = "euclidean", library= "GUDHI", printProgress = FALSE),
    error = function(e)
      tryCatch(ripsDiag(emb, maxdimension = maxdim, maxscale = maxscale,
               dist = "euclidean", printProgress = FALSE), error = function(e2) NULL)
  )
```

```
  if (is.null(pd) || is.null(pd$diagram) || nrow(pd$diagram) == 0L) return(out0)
  diagm     <- pd$diagram
  lifetimes <- diagm[, 3] - diagm[, 2]
  lif1 <- lifetimes[diagm[, 1] == 1]
  lif0 <- lifetimes[diagm[, 1] == 0]
  top1_1 <- ifelse(length(lif1) > 0L, sort(lif1, TRUE)[1], 0)
  top2_1 <- ifelse(length(lif1) > 1L, sort(lif1, TRUE)[2], 0)
  top1_0 <- ifelse(length(lif0) > 0L, sort(lif0)[length(lif0)], 0)
  L <- lifetimes[lifetimes > 0]
  pentropy <- if (length(L) == 0L) 0 else {
    p <- L / sum(L) - sum(p * log(p + 1e-12))
  }
  betti1     <- sum(diagm[, 1] == 1 & lifetimes > 0)
  total_pers <- sum(L)
  return(c(top1_H1 = top1_1,  top2_H1 = top2_1,  top1_H0 = top1_0,
           pentropy = pentropy, betti1 = betti1, total_pers = total_pers))
}
## ---- Global FPCA + derivative norms (training-based) ----
compute_fda_features_global <- function(Y_windows, N_train, nharm = NHARM_FDA) {
  N_total <- nrow(Y_windows)
  W       <- ncol(Y_windows)

  ## Training-only centering
  Y_train        <- Y_windows[1:N_train, drop = FALSE]
  col_mean       <- colMeans(Y_train, na.rm = TRUE)
  Y_center_train <- sweep(Y_train, 2, col_mean, "-")
  Y_center_all   <- sweep(Y_windows, 2, col_mean, "-")
  pc     <- prcomp(Y_center_train, center = FALSE, scale. = FALSE)
  nh_use <- min(nharm, ncol(pc$rotation))
  if (nh_use < 1L) {
    scores_all <- matrix(0, nrow = N_total, ncol = nharm)
  } else {
    loadings   <- pc$rotation[, 1:nh_use, drop = FALSE]
    scores_all <- as.matrix(Y_center_all) %*% loadings
    if (nh_use < nharm) {
      scores_all <- cbind(scores_all, matrix(0, nrow = N_total, ncol = nharm - nh_use))
    }
  }
  colnames(scores_all) <- paste0("fscore", 1:nharm)
  tGrid <- seq(0, 1, length.out = W)
  dt    <- mean(diff(tGrid))
  if (W >= 2L) {
    d1 <- t(apply(Y_windows, 1, diff)) / dt
  } else {
    d1 <- matrix(0, nrow = N_total, ncol = 1)
  }
  if ((W - 1L) >= 2L) {
    d2 <- t(apply(d1, 1, diff)) / dt
  } else {
    d2 <- matrix(0, nrow = N_total, ncol = 1)
  }
  norm_d1 <- sqrt(rowSums(d1^2) * dt)
  norm_d2 <- sqrt(rowSums(d2^2) * dt)
  fda_df <- data.frame(scores_all, norm_d1 = norm_d1, norm_d2 = norm_d2)
  return(fda_df)
}
## ---- Scaling helper ----
scale_train_test <- function(trainX, testX) {
  mu  <- apply(trainX, 2, mean)
  sdv <- apply(trainX, 2, sd)
  sdv[sdv == 0] <- 1
  return(list(train = as.data.frame(scale(trainX, center = mu, scale = sdv)),
         test = as.data.frame(scale(testX, center = mu, scale = sdv)),
         mu = mu, sd  = sdv))
}
## ---- Time-series (rolling-origin) folds ----
make_rolling_folds <- function(n, k = 5L) {
  fold_size <- floor(n / k)
  out <- vector("list", k)
  for (f in seq_len(k)) {
    val_start <- (f - 1L) * fold_size + 1L
    if (f < k) val_end <- f * fold_size else val_end <- n
    val_idx   <- val_start:val_end
    train_idx <- 1L:(val_start - 1L)
```

```
    if (length(train_idx) < 30L) next
    out[[f]] <- list(train = train_idx, val = val_idx)
  }
  return(Filter(Negate(is.null), out))
}

## ---- Threshold selection (Youden & F1) ----
find_thresholds <- function(y_true_bin, prob) {
  ok <- which(is.finite(prob) & !is.na(y_true_bin))
  y  <- y_true_bin[ok]
  p  <- prob[ok]
  thr_cand <- sort(unique(p))
  thr_cand <- sort(unique(c(0, thr_cand, 1)))
  best_J  <- -Inf; thr_J  <- 0.5
  best_F1 <- -Inf; thr_F1 <- 0.5
  for (th in thr_cand) {
    y_hat <- as.integer(p >= th)
    tp <- sum(y_hat == 1 & y == 1)
    tn <- sum(y_hat == 0 & y == 0)
    fp <- sum(y_hat == 1 & y == 0)
    fn <- sum(y_hat == 0 & y == 1)
    tpr <- ifelse(tp + fn == 0, 0, tp / (tp + fn))
    tnr <- ifelse(tn + fp == 0, 0, tn / (tn + fp))
    prec <- ifelse(tp + fp == 0, 0, tp / (tp + fp))
    rec  <- tpr
    F1   <- ifelse(prec + rec == 0, 0, 2 * prec * rec / (prec + rec))
    J    <- tpr + tnr - 1
    if (J > best_J) {
      best_J <- J
      thr_J  <- th
    }
    if (F1 > best_F1) {
      best_F1 <- F1
      thr_F1  <- th
    }
  }
  return(list(thr_J = thr_J, thr_F1 = thr_F1))
}
## ---- Metrics at given threshold ----
compute_metrics <- function(y_true_bin, prob, thr) {
  y_hat <- as.integer(prob >= thr)
  tp   <- sum(y_hat == 1 & y_true_bin == 1)
  tn   <- sum(y_hat == 0 & y_true_bin == 0)
  fp   <- sum(y_hat == 1 & y_true_bin == 0)
  fn   <- sum(y_hat == 0 & y_true_bin == 1)
  acc  <- (tp + tn) / (tp + tn + fp + fn)
  prec <- ifelse(tp + fp == 0, 0, tp / (tp + fp))
  rec  <- ifelse(tp + fn == 0, 0, tp / (tp + fn))
  F1   <- ifelse(prec + rec == 0, 0, 2 * prec * rec / (prec + rec))
    return(c(Accuracy = acc, Precision = prec, Recall = rec, F1 = F1))
}
## ---- Block-bootstrap AUC difference (model2 - model1), parallel ----
block_boot_auc_diff <- function(y_bin, p1, p2, B = N_BOOT, block_len = BLOCK_LEN) {
  n <- length(y_bin)
  stopifnot(length(p1) == n, length(p2) == n)
  y_bin <- as.integer(y_bin)
    diffs <- foreach(b = 1:B, .combine = c,
                     .packages = "pROC") %dopar% {
                       idx <- integer(0)
                       while (length(idx) < n) {
                         start <- sample(1:max(1, n - block_len + 1), 1)
                         block <- start:min(start + block_len - 1L, n)
                         idx   <- c(idx, block)
                       }
                       idx <- idx[1:n]
                       roc1 <- tryCatch(
                         pROC::roc(response = y_bin[idx], predictor = p1[idx],
                                   quiet = TRUE, direction = "<"),
                         error = function(e) NULL
                       )
                       roc2 <- tryCatch(
                         pROC::roc(response = y_bin[idx], predictor = p2[idx],
                                   quiet = TRUE, direction = "<"),
                         error = function(e) NULL
```

```
                    )
                    if (is.null(roc1) || is.null(roc2)) return(NA_real_)
                    return(as.numeric(pROC::auc(roc2) - pROC::auc(roc1)))
                  }
  diffs <- diffs[is.finite(diffs)]
  if (length(diffs) < 20L) {
    return(list(diff_hat = NA_real_, ci_low = NA_real_, ci_high  = NA_real_, p = NA_real_,
                B_eff  = length(diffs)))
  }
  diff_hat <- mean(diffs)
  ci <- quantile(diffs, c(0.025, 0.975), na.rm = TRUE)
  pval <- 2 * min(mean(diffs <= 0), mean(diffs >= 0))
    return(list(diff_hat = diff_hat, ci_low = ci[1],
                ci_high = ci[2], p = pval, B_eff = length(diffs)))
}
## ---- Block-bootstrap PR-AUC CI (parallel) ----
block_boot_pr_auc <- function(y_bin, prob, B = N_BOOT, block_len = BLOCK_LEN) {
  n <- length(y_bin)
  y_bin <- as.integer(y_bin)
  aucs <- foreach(b = 1:B, .combine = c,
                  .packages = "PRROC") %dopar% {
                    idx <- integer(0)
                    while (length(idx) < n) {
                      start <- sample(1:max(1, n - block_len + 1), 1)
                      block <- start:min(start + block_len - 1L, n)
                      idx   <- c(idx, block)
                    }
                    idx <- idx[1:n]
                    yb  <- y_bin[idx]
                    pb  <- prob[idx]
                    if (sum(yb == 1) == 0L || sum(yb == 0) == 0L) return(NA_real_)
                    prb <- tryCatch(
                      PRROC::pr.curve(scores.class0 = pb[yb == 1],
                                      scores.class1 = pb[yb == 0],
                                      curve = FALSE),
                      error = function(e) NULL
                    )
                    if (is.null(prb)) return(NA_real_)
                    return(prb$auc.integral)
                  }
  aucs <- aucs[is.finite(aucs)]
  if (length(aucs) < 20L) {
    return(list(est = NA_real_, ci_low = NA_real_, ci_high = NA_real_))
  }
  est <- mean(aucs)
  ci  <- quantile(aucs, c(0.025, 0.975), na.rm = TRUE)
  return(list(est = est, ci_low = ci[1], ci_high = ci[2]))
}
## ---- ECE from calibration table ----
compute_ece_from_calib <- function(calib_df) {
  calib_df <- calib_df[is.finite(calib_df$prob_mean) &
                         is.finite(calib_df$y_mean) &
                         calib_df$n_bin > 0, ]
  if (nrow(calib_df) == 0L) return(NA_real_)

  w <- calib_df$n_bin / sum(calib_df$n_bin)
  return(sum(w * abs(calib_df$y_mean - calib_df$prob_mean)))
}
## ---- Calibration (Brier + ECE) with fixed breaks ----
compute_calib <- function(prob_vec, y_bin, breaks) {
  brier <- mean((prob_vec - y_bin)^2)
  calib_df <- data.frame(prob = prob_vec, y  = y_bin) %>%
    dplyr::mutate(bin = cut(prob, breaks = breaks, include.lowest = TRUE, right = FALSE)) %>%
    dplyr::group_by(bin) %>%
    dplyr::summarise(
      prob_mean = mean(prob),
      y_mean  = mean(y),
      n_bin = dplyr::n(),
      .groups = "drop"
    )
  ece <- compute_ece_from_calib(calib_df)
  return(list(brier = brier, ece = ece, calib_df = calib_df))
}
## ============================================================
```

```
## 2. Build asset-level dataset (windows + TDA + global-FPCA + HAR 5/22/66)
## ============================================================
build_asset_dataset <- function(sym) {
  prc_obj <- get_prices(sym)
  if (is.null(prc_obj)) return(NULL)
  cp       <- prc_obj$close
  n_na_raw <- prc_obj$n_na_raw
  ## 2.1 Raw log returns
  r_xts   <- diff(log(cp))
  r_xts   <- na.omit(r_xts)
  r       <- as.numeric(r_xts)
  dates_r <- index(r_xts)
  n_r     <- length(r)
  ## 2.2 Candidate window anchors (need WINDOW_LEN past, K_RV future)
  idx_anchor <- seq(WINDOW_LEN, n_r - K_RV)
  N_total    <- length(idx_anchor)
  if (N_total < 200L) {
    warning("[WARN] Too few windows after constraints: ", sym)
    return(NULL)
  }
  ## 2.3 Train/test split at window level
  N_train <- floor(TRAIN_FRAC * N_total)
  idx_anchor_train <- idx_anchor[1:N_train]
  ## 2.4 Training-only returns for preprocessing
  idx_ret_train <- unique(unlist(
    lapply(idx_anchor_train, function(i)
      (i - WINDOW_LEN + 1L):(i + K_RV))
  ))
  idx_ret_train <- idx_ret_train[idx_ret_train >= 1L &
                                 idx_ret_train <= n_r]
  r_train_all   <- r[idx_ret_train]
  ## 2.5 Winsorization
  qs_r  <- quantile(r_train_all, probs = c(0.01, 0.99), na.rm = TRUE)
  r_win <- pmin(pmax(r, qs_r[1]), qs_r[2])
  abs_r_win <- abs(r_win)
  ## 2.6 Forward volatility proxy: mean |r| over next K_RV days
  rv_future <- rep(NA_real_, n_r)
  for (t in 1:(n_r - K_RV)) {
    rv_future[t] <- mean(abs_r_win[(t + 1L):(t + K_RV)])
  }
  ## 2.7 Spike threshold from training anchors only
  thr_spike <- quantile(rv_future[idx_anchor_train], probs = SPIKE_Q, na.rm = TRUE)
  ## 2.8 Embedding parameters (train-only, |r|)
  emb_par <- select_embedding_params(abs_r_win[idx_ret_train])
  m_use   <- emb_par$m
  tau_use <- emb_par$tau
  ## 2.9 Pre-allocate
  feat_tda <- matrix(0, nrow = N_total, ncol = 6)
  colnames(feat_tda) <- c("top1_H1", "top2_H1", "top1_H0", "pentropy", "betti1", "total_pers")
  Y_windows <- matrix(NA_real_, nrow = N_total, ncol = WINDOW_LEN)
  labels    <- integer(N_total)
  har_d <- har_w <- har_m <- numeric(N_total)
  message("   Windows: ", N_total,
          " | train = ", N_train,
          " | test = ", N_total - N_train)
  ## 2.10 Loop over anchors
  for (idx in seq_along(idx_anchor)) {
    i    <- idx_anchor[idx]
    w_id <- (i - WINDOW_LEN + 1L):i
    y_win <- r_win[w_id]
    Y_windows[idx, ] <- y_win
    ## TDA on |returns| in the window
    feat_tda[idx, ] <- tda_features_safe(y_window  = abs(y_win), dim_embed = m_use, tau   = tau_use)
    ## HAR-type averages from the full winsorised |r| series
    idx_d_start <- max(1L, i - 5L  + 1L)
    idx_w_start <- max(1L, i - 22L + 1L)
    idx_m_start <- max(1L, i - 66L + 1L)
    har_d[idx]  <- mean(abs_r_win[idx_d_start:i])
    har_w[idx]  <- mean(abs_r_win[idx_w_start:i])
    har_m[idx]  <- mean(abs_r_win[idx_m_start:i])
    labels[idx] <- as.integer(rv_future[i] > thr_spike)
  }
  ## 2.11 FDA features via global FPCA
  fda_df   <- compute_fda_features_global(Y_windows, N_train = N_train, nharm = NHARM_FDA)
```

```
  feat_fda <- as.matrix(fda_df)
  ## 2.12 Assemble data.frame
  df <- data.frame(Asset = sym, anchor_id = idx_anchor, date = as.Date(dates_r[idx_anchor]),
                   label  = labels, rv_future  = rv_future[idx_anchor], har_d  = har_d,
                   har_w  = har_w, har_m  = har_m, feat_tda, feat_fda, stringsAsFactors = FALSE)
  df$set <- "train"
  df$set[(N_train + 1L):N_total] <- "test"
  return(list(data = df,
              info = list(Asset = sym, n_raw = n_r, N_total = N_total, N_train = N_train,
              N_test  = N_total - N_train, winsor_lo = qs_r[1], winsor_hi = qs_r[2],
              thr_spike = as.numeric(thr_spike), emb_m = m_use,  emb_tau = tau_use,
              n_na_raw  = n_na_raw), r_win = r_win, dates_r  = dates_r,
              rv_future_full = rv_future, idx_anchor = idx_anchor))
}
## ============================================================
## 3. Models: RF (FDA/TDA/BOTH) + XGBoost (HAR) with rolling CV
## ============================================================
## ---- Rolling RF with CV and OOF thresholds ----
run_rf_ts <- function(X_train, X_test, y_train_fac, folds, feature_set_name) {
  n <- nrow(X_train)
  p <- ncol(X_train)
  grid_mtry <- sort(unique(
    pmax(1L, pmin(p, c(floor(sqrt(p)), floor(p / 3), floor(p / 2))))
  ))
  cv_res <- data.frame(mtry = integer(0), mean_auc = numeric(0), sd_auc = numeric(0))
  for (mtry_val in grid_mtry) {
    aucs <- numeric(length(folds))
    for (f_idx in seq_along(folds)) {
      f      <- folds[[f_idx]]
      tr_idx <- f$train
      va_idx <- f$val
      X_tr   <- X_train[tr_idx, drop = FALSE]
      X_va   <- X_train[va_idx, drop = FALSE]
      y_tr   <- y_train_fac[tr_idx]
      y_va   <- y_train_fac[va_idx]
      sc     <- scale_train_test(X_tr, X_va)
      rf     <- randomForest::randomForest(x = sc$train, y = y_tr, mtry = mtry_val, ntree = 500,
                                           nodesize = 10, importance = FALSE)
      prob_va <- predict(rf, sc$test, type = "prob")[, "spike"]
      roc_va  <- pROC::roc(response = y_va, predictor = prob_va, quiet = TRUE, direction = "<")
      aucs[f_idx] <- as.numeric(pROC::auc(roc_va))
    }
    cv_res <- rbind(cv_res, data.frame(mtry = mtry_val, mean_auc = mean(aucs), sd_auc = sd(aucs)))
  }
  best_mtry <- cv_res$mtry[which.max(cv_res$mean_auc)]
  ## OOF predictions (best mtry)
  oof_prob <- rep(NA_real_, n)
  for (f in folds) {
    tr_idx <- f$train
    va_idx <- f$val
    X_tr   <- X_train[tr_idx, drop = FALSE]
    X_va   <- X_train[va_idx, drop = FALSE]
    y_tr   <- y_train_fac[tr_idx]
    sc <- scale_train_test(X_tr, X_va)
    rf <- randomForest::randomForest(x = sc$train, y = y_tr, mtry = best_mtry, ntree  = 500, nodesize = 10)
    prob_va           <- predict(rf, sc$test, type = "prob")[, "spike"]
    oof_prob[va_idx]  <- prob_va
  }
  y_train_bin <- as.integer(y_train_fac == "spike")
  thr_list <- find_thresholds(y_train_bin, oof_prob)
  ## Final fit on full training set
  sc_full <- scale_train_test(X_train, X_test)
  rf_final <- randomForest::randomForest(
    x = sc_full$train, y = y_train_fac, mtry = best_mtry, ntree = 1000, nodesize = 10,
    importance = TRUE
  )
  prob_test <- predict(rf_final, sc_full$test, type = "prob")[, "spike"]
  return(list(
    model_name = feature_set_name,
    rf     = rf_final,
    mtry_best  = best_mtry,
    cv_table = cv_res,
    oof_prob = oof_prob,
    thr_J  = thr_list$thr_J,
```

```
    thr_F1 = thr_list$thr_F1,
    prob_test  = prob_test,
    scaler_mu  = sc_full$mu,
    scaler_sd  = sc_full$sd
  ))
}
## ---- Rolling XGBoost with coarse grid + OOF thresholds ----
run_xgb_ts <- function(X_train, X_test, y_train_bin, y_train_fac, folds) {
  n <- nrow(X_train)
  xgb_grid <- expand.grid(max_depth = c(2L, 3L), eta = c(0.05, 0.10), nrounds = c(200L, 400L),
                          subsample = 0.8, colsample_bytree = 0.8)
  best_mean_auc <- -Inf
  best_row <- NULL
  for (g in seq_len(nrow(xgb_grid))) {
    params_g <- list(objective = "binary:logistic", eval_metric = "auc",
                     max_depth = xgb_grid$max_depth[g], eta = xgb_grid$eta[g],
                     subsample = xgb_grid$subsample[g],
                     colsample_bytree = xgb_grid$colsample_bytree[g])
    nrounds_g <- xgb_grid$nrounds[g]
    aucs <- numeric(length(folds))
    for (f_idx in seq_along(folds)) {
      f      <- folds[[f_idx]]
      tr_idx <- f$train
      va_idx <- f$val
      dtr <- xgboost::xgb.DMatrix(
        data  = X_train[tr_idx, drop = FALSE],
        label = y_train_bin[tr_idx]
      )
      dva <- xgboost::xgb.DMatrix(
        data  = X_train[va_idx, drop = FALSE],
        label = y_train_bin[va_idx]
      )
      xgb_fit_cv <- xgboost::xgb.train(
        params  = params_g,
        data    = dtr,
        nrounds = nrounds_g,
        verbose = 0
      )
      pred_va <- predict(xgb_fit_cv, dva)
      roc_va  <- pROC::roc(response = y_train_fac[va_idx],
                           predictor = pred_va,
                           quiet = TRUE, direction = "<")
      aucs[f_idx] <- as.numeric(pROC::auc(roc_va))
    }
    mean_auc <- mean(aucs)
    if (mean_auc > best_mean_auc) {
      best_mean_auc <- mean_auc
      best_row <- list(max_depth = xgb_grid$max_depth[g], eta = xgb_grid$eta[g],
                       nrounds = nrounds_g, subsample = xgb_grid$subsample[g],
                       colsample_bytree = xgb_grid$colsample_bytree[g],
                       mean_auc = mean_auc,  sd_auc = sd(aucs))
    }
  }
  best_params <- list(objective = "binary:logistic", eval_metric = "auc",
                      max_depth = best_row$max_depth, eta = best_row$eta,
                      subsample = best_row$subsample,
                      colsample_bytree = best_row$colsample_bytree)
  best_nrounds <- best_row$nrounds
  ## OOF predictions
  oof_prob <- rep(NA_real_, n)
  for (f in folds) {
    tr_idx <- f$train
    va_idx <- f$val
    dtr <- xgboost::xgb.DMatrix(
      data  = X_train[tr_idx, drop = FALSE],
      label = y_train_bin[tr_idx]
    )
    dva <- xgboost::xgb.DMatrix(
      data  = X_train[va_idx, drop = FALSE],
      label = y_train_bin[va_idx]
    )
    xgb_fit_fold <- xgboost::xgb.train(
      params  = best_params,
      data  = dtr,
```

```
      nrounds = best_nrounds,
      verbose = 0
    )
    oof_prob[va_idx] <- predict(xgb_fit_fold, dva)
  }
  thr_list <- find_thresholds(y_train_bin, oof_prob)
  ## Final fit
  dtrain <- xgboost::xgb.DMatrix(data = X_train, label = y_train_bin)
  dtest  <- xgboost::xgb.DMatrix(data = X_test)
  xgb_final <- xgboost::xgb.train(
    params  = best_params,
    data  = dtrain,
    nrounds = best_nrounds,
    verbose = 0
  )
  prob_test <- as.numeric(predict(xgb_final, dtest))
  return(list(
      model_name = "xgb",
      xgb        = xgb_final,
      cv_table   = data.frame(max_depth = best_row$max_depth, eta   = best_row$eta,
                              nrounds = best_row$nrounds, mean_auc  = best_row$mean_auc,
                              sd_auc  = best_row$sd_auc),
      oof_prob = oof_prob,
      thr_J  = thr_list$thr_J,
      thr_F1 = thr_list$thr_F1,
      prob_test  = prob_test,
      params = best_params,
      nrounds  = best_nrounds
  ))
}
## ---- Fit all models for one asset (including calib & tests) ----
fit_models_for_asset <- function(asset_obj) {
  df   <- asset_obj$data
  sym  <- unique(df$Asset)
  info <- asset_obj$info
  train_df <- df[df$set == "train", ]
  test_df  <- df[df$set == "test", ]
  y_train <- train_df$label
  y_test  <- test_df$label
  y_train_fac <- factor(ifelse(y_train == 1, "spike", "normal"),
                        levels = c("normal", "spike"))
  y_test_fac  <- factor(ifelse(y_test  == 1, "spike", "normal"),
                        levels = c("normal", "spike"))
  y_train_bin <- as.integer(y_train_fac == "spike")
  y_test_bin  <- as.integer(y_test_fac  == "spike")
  cols_tda <- c("top1_H1", "top2_H1", "top1_H0",
                "pentropy", "betti1", "total_pers")
  cols_fda     <- c(paste0("fscore", 1:NHARM_FDA), "norm_d1", "norm_d2")
  cols_har     <- c("har_d", "har_w", "har_m")
  X_train_tda  <- as.matrix(train_df[, cols_tda, drop = FALSE])
  X_train_fda  <- as.matrix(train_df[, cols_fda, drop = FALSE])
  X_train_both <- as.matrix(train_df[, c(cols_tda, cols_fda, cols_har), drop = FALSE])
  X_train_xgb  <- as.matrix(train_df[, cols_har, drop = FALSE])
  X_test_tda   <- as.matrix(test_df[, cols_tda, drop = FALSE])
  X_test_fda   <- as.matrix(test_df[, cols_fda, drop = FALSE])
  X_test_both  <- as.matrix(test_df[, c(cols_tda, cols_fda, cols_har), drop = FALSE])
  X_test_xgb   <- as.matrix(test_df[, cols_har, drop = FALSE])
  folds        <- make_rolling_folds(length(y_train), k = 5L)
  ## --- RF Models ---
  mod_fda <- run_rf_ts(X_train = X_train_fda, X_test = X_test_fda,
                       y_train_fac = y_train_fac, folds = folds, feature_set_name = "fda")
  mod_tda <- run_rf_ts(X_train = X_train_tda, X_test = X_test_tda,
                       y_train_fac = y_train_fac, folds = folds, feature_set_name = "tda")
  mod_both <- run_rf_ts(X_train = X_train_both, X_test = X_test_both,
                        y_train_fac = y_train_fac, folds = folds, feature_set_name = "both")
  ## --- XGBoost (HAR benchmark) ---
  mod_xgb <- run_xgb_ts(X_train = X_train_xgb, X_test = X_test_xgb,
                        y_train_bin = y_train_bin, y_train_fac = y_train_fac, folds = folds)
  ## --- Thresholds for test set metrics (based on OOF F1-opt) ---
  thr_fda  <- mod_fda$thr_F1
  thr_tda  <- mod_tda$thr_F1
  thr_both <- mod_both$thr_F1
  thr_xgb  <- mod_xgb$thr_F1
  met_fda  <- compute_metrics(y_test_bin, mod_fda$prob_test, thr_fda)
```

```
met_tda  <- compute_metrics(y_test_bin, mod_tda$prob_test, thr_tda)
met_both <- compute_metrics(y_test_bin, mod_both$prob_test, thr_both)
met_xgb  <- compute_metrics(y_test_bin, mod_xgb$prob_test, thr_xgb)
## --- Calibration breaks (based on ALL OOF predictions) ---
oof_ref <- c(mod_fda$oof_prob, mod_tda$oof_prob, mod_both$oof_prob)
oof_ref <- oof_ref[is.finite(oof_ref)]
prob_breaks <- quantile(oof_ref, probs = seq(0, 1, length.out = B_BIN + 1), na.rm = TRUE)
prob_breaks[1] <- 0
prob_breaks[length(prob_breaks)] <- 1
## --- Calibration on test set ---
calib_fda  <- compute_calib(mod_fda$prob_test,  y_test_bin, prob_breaks)
calib_tda  <- compute_calib(mod_tda$prob_test,  y_test_bin, prob_breaks)
calib_both <- compute_calib(mod_both$prob_test, y_test_bin, prob_breaks)
calib_xgb  <- compute_calib(mod_xgb$prob_test,  y_test_bin, prob_breaks)
## --- ROC / PR metrics on test set ---
prob_fda  <- mod_fda$prob_test
prob_tda  <- mod_tda$prob_test
prob_both <- mod_both$prob_test
prob_xgb  <- mod_xgb$prob_test
roc_fda   <- pROC::roc(y_test_fac, prob_fda, quiet = TRUE, direction = "<")
roc_tda   <- pROC::roc(y_test_fac, prob_tda, quiet = TRUE, direction = "<")
roc_both  <- pROC::roc(y_test_fac, prob_both, quiet = TRUE, direction = "<")
roc_xgb   <- pROC::roc(y_test_fac, prob_xgb, quiet = TRUE, direction = "<")
auc_fda   <- as.numeric(pROC::auc(roc_fda))
auc_tda   <- as.numeric(pROC::auc(roc_tda))
auc_both  <- as.numeric(pROC::auc(roc_both))
auc_xgb   <- as.numeric(pROC::auc(roc_xgb))
ci_fda    <- as.numeric(pROC::ci.auc(roc_fda, method = "delong"))
ci_tda    <- as.numeric(pROC::ci.auc(roc_tda, method = "delong"))
ci_both   <- as.numeric(pROC::ci.auc(roc_both, method = "delong"))
ci_xgb    <- as.numeric(pROC::ci.auc(roc_xgb, method = "delong"))
## PR-AUC + block-bootstrap CIs
pr_fda <- PRROC::pr.curve(scores.class0 = prob_fda[y_test_bin == 1],
                          scores.class1 = prob_fda[y_test_bin == 0], curve = FALSE)
pr_tda <- PRROC::pr.curve(scores.class0 = prob_tda[y_test_bin == 1],
                          scores.class1 = prob_tda[y_test_bin == 0], curve = FALSE)
pr_both <- PRROC::pr.curve(scores.class0 = prob_both[y_test_bin == 1],
                           scores.class1 = prob_both[y_test_bin == 0], curve = FALSE)
pr_xgb <- PRROC::pr.curve(scores.class0 = prob_xgb[y_test_bin == 1],
                          scores.class1 = prob_xgb[y_test_bin == 0], curve = FALSE)
bb_pr_fda  <- block_boot_pr_auc(y_test_bin, prob_fda)
bb_pr_tda  <- block_boot_pr_auc(y_test_bin, prob_tda)
bb_pr_both <- block_boot_pr_auc(y_test_bin, prob_both)
bb_pr_xgb  <- block_boot_pr_auc(y_test_bin, prob_xgb)
## AUC differences (DeLong)
del_fda_tda  <- pROC::roc.test(roc_fda, roc_tda, method = "delong",
                               alternative = "two.sided", asymptotic = TRUE, quiet = TRUE)
del_tda_both <- pROC::roc.test(roc_tda, roc_both, method = "delong",
                               alternative = "two.sided", asymptotic = TRUE, quiet = TRUE)
del_fda_both <- pROC::roc.test(roc_fda, roc_both, method = "delong",
                               alternative = "two.sided", asymptotic = TRUE, quiet = TRUE)
del_xgb_both <- pROC::roc.test(roc_xgb, roc_both, method = "delong",
                               alternative = "two.sided", asymptotic = TRUE, quiet = TRUE)
## AUC differences (Block Bootstrap)
bb_fda_tda  <- block_boot_auc_diff(y_test_bin, prob_fda, prob_tda)
bb_tda_both <- block_boot_auc_diff(y_test_bin, prob_tda, prob_both)
bb_fda_both <- block_boot_auc_diff(y_test_bin, prob_fda, prob_both)
bb_xgb_both <- block_boot_auc_diff(y_test_bin, prob_xgb, prob_both)
return(list(Asset = sym, y_test_bin = y_test_bin,
            prob_fda = prob_fda, prob_tda = prob_tda, prob_both = prob_both, prob_xgb = prob_xgb,
            roc_fda = roc_fda, roc_tda = roc_tda, roc_both = roc_both, roc_xgb = roc_xgb,
            aucs = list(fda = list(auc = auc_fda, ci = ci_fda), tda = list(auc = auc_tda, ci = ci_tda),
                        both = list(auc = auc_both, ci = ci_both), xgb = list(auc = auc_xgb, ci = ci_xgb)),
  pr_aucs = list(
    fda = list(est = pr_fda$auc.integral, ci_low = bb_pr_fda$ci_low, ci_high = bb_pr_fda$ci_high),
    tda = list(est = pr_tda$auc.integral, ci_low = bb_pr_tda$ci_low, ci_high = bb_pr_tda$ci_high),
    both = list(est = pr_both$auc.integral, ci_low = bb_pr_both$ci_low, ci_high = bb_pr_both$ci_high),
    xgb = list(est = pr_xgb$auc.integral, ci_low = bb_pr_xgb$ci_low, ci_high = bb_pr_xgb$ci_high)),
    metrics_F1 = list(fda = c(Threshold = thr_fda, met_fda), tda = c(Threshold = thr_tda, met_tda),
                      both = c(Threshold = thr_both, met_both), xgb = c(Threshold = thr_xgb, met_xgb)),
    delong = list(fda_vs_tda = del_fda_tda, tda_vs_both = del_tda_both,
                  fda_vs_both = del_fda_both, xgb_vs_both = del_xgb_both),
    bb = list(fda_vs_tda = bb_fda_tda, tda_vs_both = bb_tda_both,
              fda_vs_both = bb_fda_both, xgb_vs_both = bb_xgb_both),
```

```
      calib = list(fda = calib_fda, tda = calib_tda, both = calib_both, xgb = calib_xgb),
      models = list(fda = mod_fda, tda = mod_tda, both = mod_both, xgb = mod_xgb),
      calib_breaks = prob_breaks
  ))
}

## ============================================================
## 4. Descriptive tables & plots
## ============================================================
## ---- Feature descriptives ----
compute_feature_descriptives <- function(asset_obj) {
  df <- asset_obj$data
  sym <- unique(df$Asset)
  cols_all <- c("har_d", "har_w", "har_m", "top1_H1", "top2_H1", "top1_H0",  "pentropy", "betti1",
                "total_pers", paste0("fscore", 1:NHARM_FDA), "norm_d1", "norm_d2")
  res_list <- lapply(cols_all, function(col) {
    x_train <- df[df$set == "train", col]
    x_test  <- df[df$set == "test", col]
    data.frame(
      Asset = sym,
      Feature = col,
      Mean_train = mean(x_train, na.rm = TRUE),
      SD_train = sd(x_train, na.rm = TRUE),
      Skewness_train = moments::skewness(x_train, na.rm = TRUE),
      Kurtosis_train = moments::kurtosis(x_train, na.rm = TRUE),
      Mean_test = mean(x_test, na.rm = TRUE),
      SD_test = sd(x_test, na.rm = TRUE),
      stringsAsFactors = FALSE
    )
  })
  return(do.call(rbind, res_list))
}
## ---- Feature boxplot for fscore1/har_d by class ----
plot_feature_by_class <- function(asset_res) {
  sym <- asset_res$Asset
  df <- asset_res$models$both$rf$model %>%
    dplyr::select(label, fscore1, har_d) %>%
    dplyr::mutate(label = factor(label))
  p1 <- ggplot2::ggplot(df, aes(x = factor(label), y = har_d, fill = factor(label))) +
    ggplot2::geom_violin(alpha = 0.4) +
    ggplot2::geom_boxplot(width = 0.15, outlier.size = 0.5) +
    ggplot2::scale_x_discrete(labels = c("normal", "spike")) +
    ggplot2::labs(x = "Class", y = "HAR daily (har_d)", title = paste0(sym, " - HAR: har_d by class")) +
    ggplot2::theme_minimal() +
    ggplot2::theme(legend.position = "none")
  p2 <- ggplot2::ggplot(df, aes(x = factor(label), y = fscore1, fill = factor(label))) +
    ggplot2::geom_violin(alpha = 0.4) +
    ggplot2::geom_boxplot(width = 0.15, outlier.size = 0.5) +
    ggplot2::scale_x_discrete(labels = c("normal", "spike")) +
    ggplot2::labs(x = "Class", y = "FPCA score 1", title = paste0(sym, " - FDA: fscore1 by class")) +
    ggplot2::theme_minimal() +
    ggplot2::theme(legend.position = "none")
  return(cowplot::plot_grid(p1, p2, ncol = 2))
}
## ---- ROC plot for RF + XGBoost ----
plot_roc_four <- function(asset_res) {
  sym <- asset_res$Asset
  roc_fda <- asset_res$roc_fda
  roc_tda <- asset_res$roc_tda
  roc_both <- asset_res$roc_both
  roc_xgb <- asset_res$roc_xgb
  df_roc <- rbind(
    data.frame(model = "FDA", TPR = rev(roc_fda$sensitivities), FPR = rev(1 - roc_fda$specificities)),
    data.frame(model = "TDA", TPR = rev(roc_tda$sensitivities), FPR = rev(1 - roc_tda$specificities)),
    data.frame(model = "BOTH", TPR = rev(roc_both$sensitivities), FPR = rev(1 - roc_both$specificities)),
    data.frame(model = "XGBoost", TPR = rev(roc_xgb$sensitivities), FPR = rev(1 - roc_xgb$specificities))
  )
  p_roc <- ggplot2::ggplot(df_roc, aes(x = FPR, y = TPR, color = model)) +
    ggplot2::geom_line(linewidth = 1) +
    ggplot2::geom_abline(slope = 1, intercept = 0, linetype = "dashed") +
    ggplot2::coord_equal() +
    ggplot2::labs(title = paste0(sym, " - ROC curves (test set)"),
                  x = "False positive rate", y = "True positive rate") +
    ggplot2::theme_minimal()
```

```
  return(p_roc)
}
## ---- Calibration plot for BOTH ----
plot_calibration <- function(asset_res) {
  sym <- asset_res$Asset
  calib_df <- asset_res$calib$both$calib_df
  p_calib <- ggplot2::ggplot(calib_df, aes(x = prob_mean, y = y_mean)) +
    ggplot2::geom_abline(slope = 1, intercept = 0, linetype = "dashed") +
    ggplot2::geom_point() +
    ggplot2::geom_line() +
    ggplot2::labs(title = paste0(sym, " - Calibration (BOTH, test)"),
                  x = "Predicted probability (bin mean)",
                  y = "Observed spike frequency") +
    ggplot2::theme_minimal()
  return(p_calib)
}
## ============================================================
## 5. Main loop over assets (empirical study)
## ============================================================
asset_objects <- list()
results_models <- list()
desc_list <- list()
study_rows <- list()
cv_rows <- list()
perf_rows <- list()
auc_rows <- list()
emb_rows <- list()
calib_rows <- list()
for (sym in ASSETS) {
  ao <- build_asset_dataset(sym)
  if (is.null(ao)) next
  asset_objects[[sym]] <- ao
  info <- ao$info
  df   <- ao$data
  train_df <- df[df$set == "train", ]
  test_df  <- df[df$set == "test", ]
  ## Study sample summary
  study_rows[[sym]] <- data.frame(
    Asset = sym, Total_Days = info$n_raw,
    Start_Date = min(df$date), End_Date = max(df$date),
    N_Windows_Total = info$N_total, N_Windows_Train = info$N_train,
    N_Windows_Test = info$N_test,
    Spike_Ratio_train = mean(train_df$label == 1), Spike_Count_test = sum(test_df$label == 1),
    Normal_Count_test = sum(test_df$label == 0), Spike_Ratio_test = mean(test_df$label == 1))
  ## Embedding / winsorization summary
  emb_rows[[sym]] <- data.frame(
    Asset = sym, m = info$emb_m, tau = info$emb_tau,
    winsor_lo = info$winsor_lo, winsor_hi = info$winsor_hi, thr_spike = info$thr_spike)
  desc_list[[sym]] <- compute_feature_descriptives(ao)
  res <- fit_models_for_asset(ao)
  results_models[[sym]] <- res
  ## CV summary (RF only)
  for (mname in c("fda", "tda", "both")) {
    mod <- res$models[[mname]]
    cv_table <- mod$cv_table
    cv_best <- cv_table[which.max(cv_table$mean_auc), ]
    cv_rows[[paste(sym, mname, sep = "_")]] <- data.frame(
      Asset = sym, ModelName = mname, mtry = mod$mtry_best,
      CV_ROC_AUC_mean = cv_best$mean_auc,
      CV_ROC_AUC_sd = cv_best$sd_auc)
  }
  N_test <- length(res$y_test_bin)
  pm <- res$pr_aucs
  mf <- res$metrics_F1
  cal <- res$calib
  ## Performance table
  perf_rows[[sym]] <- rbind(
    data.frame(
      Asset = sym, Model = "FDA", N_test = N_test,
      AUC = res$aucs$fda$auc, CI_low = res$aucs$fda$ci[1],
      CI_high = res$aucs$fda$ci[3],
      PR_AUC = pm$fda$est, PR_CI_low = pm$fda$ci_low,
      PR_CI_high = pm$fda$ci_high, t(mf$fda)),
    data.frame(
```

```
      Asset = sym, Model = "TDA", N_test = N_test,
      AUC = res$aucs$tda$auc, CI_low = res$aucs$tda$ci[1],
      CI_high = res$aucs$tda$ci[3],
      PR_AUC = pm$tda$est, PR_CI_low = pm$tda$ci_low,
      PR_CI_high = pm$tda$ci_high,  t(mf$tda)),

    data.frame(
      Asset = sym, Model = "BOTH", N_test = N_test,
      AUC = res$aucs$both$auc, CI_low = res$aucs$both$ci[1],
      CI_high = res$aucs$both$ci[3],
      PR_AUC = pm$both$est, PR_CI_low = pm$both$ci_low,
      PR_CI_high = pm$both$ci_high, t(mf$both)),
    data.frame(
      Asset = sym, Model = "XGBoost", N_test = N_test,
      AUC = res$aucs$xgb$auc, CI_low = res$aucs$xgb$ci[1],
      CI_high = res$aucs$xgb$ci[3],
      PR_AUC = pm$xgb$est, PR_CI_low = pm$xgb$ci_low,
      PR_CI_high = pm$xgb$ci_high, t(mf$xgb)))
  bb_dat <- res$bb
  ## AUC comparison table
  auc_rows[[sym]] <- rbind(
    data.frame(
      Asset = sym, Comparison = "FDA_vs_TDA",
      AUC_Diff_DeLong = res$delong$fda_vs_tda$estimate,
      Pval_DeLong = res$delong$fda_vs_tda$p.value,
      AUC_Diff_BB = bb_dat$fda_vs_tda$diff_hat,
      BB_CI_low = bb_dat$fda_vs_tda$ci_low,
      BB_CI_high = bb_dat$fda_vs_tda$ci_high,
      BB_Pval = bb_dat$fda_vs_tda$p,
      BB_B_eff = bb_dat$fda_vs_tda$B_eff),
    data.frame(
      Asset = sym, Comparison = "TDA_vs_BOTH",
      AUC_Diff_DeLong = res$delong$tda_vs_both$estimate,
      Pval_DeLong = res$delong$tda_vs_both$p.value,
      AUC_Diff_BB = bb_dat$tda_vs_both$diff_hat,
      BB_CI_low = bb_dat$tda_vs_both$ci_low,
      BB_CI_high = bb_dat$tda_vs_both$ci_high,
      BB_Pval = bb_dat$tda_vs_both$p,
      BB_B_eff = bb_dat$tda_vs_both$B_eff),
    data.frame(
      Asset = sym, Comparison = "FDA_vs_BOTH",
      AUC_Diff_DeLong = res$delong$fda_vs_both$estimate,
      Pval_DeLong = res$delong$fda_vs_both$p.value,
      AUC_Diff_BB = bb_dat$fda_vs_both$diff_hat,
      BB_CI_low = bb_dat$fda_vs_both$ci_low,
      BB_CI_high = bb_dat$fda_vs_both$ci_high,
      BB_Pval = bb_dat$fda_vs_both$p,
      BB_B_eff = bb_dat$fda_vs_both$B_eff),
    data.frame(
      Asset = sym, Comparison = "XGB_vs_BOTH",
      AUC_Diff_DeLong = res$delong$xgb_vs_both$estimate,
      Pval_DeLong = res$delong$xgb_vs_both$p.value,
      AUC_Diff_BB = bb_dat$xgb_vs_both$diff_hat,
      BB_CI_low = bb_dat$xgb_vs_both$ci_low,
      BB_CI_high = bb_dat$xgb_vs_both$ci_high,
      BB_Pval = bb_dat$xgb_vs_both$p,
      BB_B_eff = bb_dat$xgb_vs_both$B_eff))
  ## Calibration table
  calib_rows[[sym]] <- rbind(
    data.frame(Asset = sym, Model = "FDA", Brier = cal$fda$brier, ECE = cal$fda$ece),
    data.frame(Asset = sym, Model = "TDA",  Brier = cal$tda$brier, ECE = cal$tda$ece),
    data.frame(Asset = sym, Model = "BOTH", Brier = cal$both$brier, ECE = cal$both$ece),
    data.frame(Asset = sym, Model = "XGBoost", Brier = cal$xgb$brier, ECE = cal$xgb$ece))
  ## Plots
  g_roc <- plot_roc_four(res)
  g_cal <- plot_calibration(res)
  g_feat <- plot_feature_by_class(res)
  ggplot2::ggsave(file.path(PLOTDIR, paste0("Fig_ROC_", sym, ".png")), g_roc,
                  width = 7, height = 5, dpi = 300)
  ggplot2::ggsave(file.path(PLOTDIR, paste0("Fig_Calibration_", sym, ".png")), g_cal,
                  width = 7, height = 5, dpi = 300)
  ggplot2::ggsave(file.path(PLOTDIR, paste0("Fig_FeatureSummaries_", sym, ".png")), g_feat,
                  width = 8, height = 4, dpi = 300)
}
```

```
## ---- Write empirical tables ----
Table_Study_Sample <- do.call(rbind, study_rows)
write.csv(Table_Study_Sample, file = file.path(TABDIR, "Table_Study_Sample.csv"), row.names = FALSE)
Table_Embedding_Params <- do.call(rbind, emb_rows)
write.csv(Table_Embedding_Params, file = file.path(TABDIR, "Table_Embedding_Params.csv"),
          row.names = FALSE)
Table_Feature_Descriptives <- do.call(rbind, desc_list)
write.csv(Table_Feature_Descriptives, file = file.path(TABDIR, "Table_Feature_Descriptives.csv"),
          row.names = FALSE)
Table_RandomForest_CV <- do.call(rbind, cv_rows)
write.csv(Table_RandomForest_CV, file = file.path(TABDIR, "Table_RandomForest_CV.csv"), row.names = FALSE)
Table_Main_Performance <- do.call(rbind, perf_rows)
write.csv(Table_Main_Performance, file = file.path(TABDIR, "Table_Main_Performance.csv"), row.names = FALSE)
Table_AUC_Comparisons <- do.call(rbind, auc_rows)
write.csv(Table_AUC_Comparisons, file = file.path(TABDIR, "Table_AUC_Comparisons.csv"), row.names = FALSE)
Table_Calibration <- do.call(rbind, calib_rows)
write.csv(Table_Calibration, file = file.path(TABDIR, "Table_Calibration.csv"), row.names = FALSE)
message("Real-data method finished. Tables in ", TABDIR, " | Plots in ", PLOTDIR)
## ============================================================
## 6. Regime-GARCH simulation with Student-t innovations
## and same spike label as in empirical study
## ============================================================
## ---- Regime-GARCH(1,1) simulator ----
simulate_regime_garch_series <- function(N = 6000L, par = NULL) {
  if (is.null(par)) {
    par <- c(omega_1 = 1e-06, alpha_1 = 0.05, beta_1 = 0.85, nu_1 = 5,omega_2 = 1e-05,
             alpha_2 = 0.08, beta_2 = 0.80, nu_2 = 10,p_11 = 0.99, p_22 = 0.90)
  }
  if (N <= 3L) stop("N must be > 3")
  omega <- c(par["omega_1"], par["omega_2"])
  alpha <- c(par["alpha_1"], par["alpha_2"])
  beta  <- c(par["beta_1"], par["beta_2"])
  nu    <- c(par["nu_1"], par["nu_2"])
  P     <- matrix(c(par["p_11"], 1 - par["p_11"], 1 - par["p_22"], par["p_22"]), nrow = 2, byrow = TRUE)
  state <- rep(1L, N)
  r     <- rep(NA_real_, N)
  sig2  <- rep(NA_real_, N)
  state[1] <- 1L # Start in regime 1
  # Initialize variance and return
  sig2[1] <- omega[state[1]] / (1 - alpha[state[1]] - beta[state[1]])
  r[1]    <- sqrt(sig2[1]) * rt(1, df = nu[state[1]])
  for (t in 2:N) {
    # Regime transition
    state[t] <- sample(1:2, 1, prob = P[state[t - 1], ])
    s        <- state[t]
    # GARCH update
    sig2[t] <- omega[s] + alpha[s] * r[t - 1]^2 + beta[s] * sig2[t - 1]
    # Standardized innovation (Student-t)
    z <- rt(1, df = nu[s])
    # Return
    r[t] <- sqrt(sig2[t]) * z
  }
  return(list(r = r, state = state))
}
run_simulation_scenario <- function(scen_name = "RegimeGARCH", N = 6000L) {
  sim      <- simulate_regime_garch_series(N = N)
  r        <- sim$r
  true_reg <- sim$state
  dates_r  <- seq.Date(from = as.Date("2000-01-01"), by = "day", length.out = length(r))
  n_r      <- length(r)
  ## Window creation/anchors (same as empirical)
  idx_anchor <- seq(WINDOW_LEN, n_r - K_RV)
  N_total <- length(idx_anchor)
  N_train <- floor(TRAIN_FRAC * N_total)
  idx_anchor_train <- idx_anchor[1:N_train]

  ## Training-only returns for preprocessing
  idx_ret_train <- unique(unlistlapply(idx_anchor_train, function(i) (i - WINDOW_LEN + 1L):(i + K_RV))))
  idx_ret_train <- idx_ret_train[idx_ret_train >= 1L & idx_ret_train <= n_r]
  r_train_all   <- r[idx_ret_train]
  ## Winsorization
  qs_r  <- quantile(r_train_all, probs = c(0.01, 0.99), na.rm = TRUE)
  r_win <- pmin(pmax(r, qs_r[1]), qs_r[2])
  abs_r_win <- abs(r_win)
```

```
  ## Forward volatility proxy & Spike threshold
  rv_future <- rep(NA_real_, n_r)
  for (t in 1:(n_r - K_RV)) {
    rv_future[t] <- mean(abs_r_win[(t + 1L):(t + K_RV)])
  }
  thr_spike <- quantile(rv_future[idx_anchor_train], probs = SPIKE_Q, na.rm = TRUE)
  ## Embedding parameters
  emb_par <- select_embedding_params(abs_r_win[idx_ret_train])
  m_use   <- emb_par$m
  tau_use <- emb_par$tau
  ## Pre-allocate
  feat_tda <- matrix(0, nrow = N_total, ncol = 6)
  colnames(feat_tda) <- c("top1_H1", "top2_H1", "top1_H0", "pentropy", "betti1", "total_pers")
  Y_windows <- matrix(NA_real_, nrow = N_total, ncol = WINDOW_LEN)
  har_d <- har_w <- har_m <- numeric(N_total)
  labels <- integer(N_total)
  message("[SIM] Scenario = ", scen_name, " | Windows: ", N_total,
          " | train = ", N_train, " | test = ", N_total - N_train)
  ## Loop over anchors
  for (idx in seq_along(idx_anchor)) {
    i    <- idx_anchor[idx]
    w_id <- (i - WINDOW_LEN + 1L):i
    y_win <- r_win[w_id]
    Y_windows[idx, ] <- y_win
    ## TDA on |returns| in the window
    feat_tda[idx, ] <- tda_features_safe(y_window = abs(y_win), dim_embed = m_use, tau = tau_use)
    ## HAR-type averages
    idx_d_start <- max(1L, i - 5L  + 1L)
    idx_w_start <- max(1L, i - 22L + 1L)
    idx_m_start <- max(1L, i - 66L + 1L)
    har_d[idx]  <- mean(abs_r_win[idx_d_start:i])
    har_w[idx]  <- mean(abs_r_win[idx_w_start:i])
    har_m[idx]  <- mean(abs_r_win[idx_m_start:i])
    labels[idx] <- as.integer(rv_future[i] > thr_spike)
  }
  ## FDA features via global FPCA
  fda_df   <- compute_fda_features_global(Y_windows, N_train = N_train, nharm = NHARM_FDA)
  ## Assemble data.frame
  sim_df <- data.frame(
    Asset  = scen_name, anchor_id  = idx_anchor,
    date   = dates_r[idx_anchor], label  = labels,
    rv_future  = rv_future[idx_anchor], har_d  = har_d,
    har_w  = har_w, har_m  = har_m,
    feat_tda, fda_df, stringsAsFactors = FALSE)
  sim_df$set <- "train"
  sim_df$set[(N_train + 1L):N_total] <- "test"
  info_list <- list(
    Asset = scen_name, n_raw = n_r,
    N_total = N_total, N_train = N_train, N_test  = N_total - N_train,
    winsor_lo = qs_r[1], winsor_hi = qs_r[2],
    thr_spike = as.numeric(thr_spike),
    emb_m = m_use, emb_tau = tau_use, n_na_raw  = 0L)
  return(list(
    data = sim_df, info = info_list, r_win = r_win, dates_r = dates_r,
    rv_future_full = rv_future, idx_anchor = idx_anchor))
}
## ---- Run simulation, tables & plots ----
sim_obj <- run_simulation_scenario("RegimeGARCH")
sim_res <- fit_models_for_asset(sim_obj)
N_test_sim <- length(sim_res$y_test_bin)
pm_sim <- sim_res$pr_aucs
mf_sim <- sim_res$metrics_F1
del_sim <- sim_res$delong
bb_sim <- sim_res$bb
cal_sim <- sim_res$calib
## Performance table
Table_Sim_Performance <- rbind(
  data.frame(
    Scenario = "RegimeGARCH", Model = "FDA", N_test = N_test_sim,
    AUC = sim_res$aucs$fda$auc, CI_low = sim_res$aucs$fda$ci[1],
    CI_high = sim_res$aucs$fda$ci[3],
    PR_AUC = pm_sim$fda$est, PR_CI_low = pm_sim$fda$ci_low,
    PR_CI_high = pm_sim$fda$ci_high, t(mf_sim$fda)),
  data.frame(
```

```
    Scenario = "RegimeGARCH", Model = "TDA", N_test = N_test_sim,
    AUC = sim_res$aucs$tda$auc, CI_low = sim_res$aucs$tda$ci[1],
    CI_high = sim_res$aucs$tda$ci[3],
    PR_AUC = pm_sim$tda$est, PR_CI_low = pm_sim$tda$ci_low,
    PR_CI_high = pm_sim$tda$ci_high, t(mf_sim$tda)),

  data.frame(
    Scenario = "RegimeGARCH", Model = "BOTH", N_test = N_test_sim,
    AUC = sim_res$aucs$both$auc, CI_low = sim_res$aucs$both$ci[1],
    CI_high = sim_res$aucs$both$ci[3],
    PR_AUC = pm_sim$both$est, PR_CI_low = pm_sim$both$ci_low,
    PR_CI_high = pm_sim$both$ci_high,  t(mf_sim$both)),
  data.frame(
    Scenario = "RegimeGARCH", Model = "XGBoost", N_test = N_test_sim,
    AUC = sim_res$aucs$xgb$auc, CI_low = sim_res$aucs$xgb$ci[1],
    CI_high = sim_res$aucs$xgb$ci[3],
    PR_AUC = pm_sim$xgb$est, PR_CI_low = pm_sim$xgb$ci_low,
    PR_CI_high = pm_sim$xgb$ci_high, t(mf_sim$xgb)))
## AUC comparison table
Table_Sim_AUC_Comparisons <- rbind(
  data.frame(
    Scenario = "RegimeGARCH", Comparison = "FDA_vs_TDA",
    AUC_Diff_DeLong = del_sim$fda_vs_tda$estimate,
    Pval_DeLong = del_sim$fda_vs_tda$p.value, AUC_Diff_BB = bb_sim$fda_vs_tda$diff_hat,
    BB_CI_low = bb_sim$fda_vs_tda$ci_low, BB_CI_high = bb_sim$fda_vs_tda$ci_high,
    BB_Pval = bb_sim$fda_vs_tda$p, BB_B_eff = bb_sim$fda_vs_tda$B_eff),
  data.frame(
    Scenario = "RegimeGARCH", Comparison = "TDA_vs_BOTH",
    AUC_Diff_DeLong = del_sim$tda_vs_both$estimate,
    Pval_DeLong = del_sim$tda_vs_both$p.value, AUC_Diff_BB = bb_sim$tda_vs_both$diff_hat,
    BB_CI_low = bb_sim$tda_vs_both$ci_low, BB_CI_high = bb_sim$tda_vs_both$ci_high,
    BB_Pval = bb_sim$tda_vs_both$p, BB_B_eff = bb_sim$tda_vs_both$B_eff),
  data.frame(
    Scenario = "RegimeGARCH", Comparison = "FDA_vs_BOTH",
    AUC_Diff_DeLong = del_sim$fda_vs_both$estimate,
    Pval_DeLong = del_sim$fda_vs_both$p.value,
    AUC_Diff_BB = bb_sim$fda_vs_both$diff_hat, BB_CI_low = bb_sim$fda_vs_both$ci_low,
    BB_CI_high = bb_sim$fda_vs_both$ci_high,
    BB_Pval = bb_sim$fda_vs_both$p, BB_B_eff = bb_sim$fda_vs_both$B_eff),
  data.frame(
    Scenario = "RegimeGARCH", Comparison = "XGB_vs_BOTH",
    AUC_Diff_DeLong = del_sim$xgb_vs_both$estimate,
    Pval_DeLong = del_sim$xgb_vs_both$p.value, AUC_Diff_BB = bb_sim$xgb_vs_both$diff_hat,
    BB_CI_low = bb_sim$xgb_vs_both$ci_low, BB_CI_high = bb_sim$xgb_vs_both$ci_high,
    BB_Pval = bb_sim$xgb_vs_both$p, BB_B_eff = bb_sim$xgb_vs_both$B_eff))
## Calibration table
Table_Sim_Calibration <- rbind(
  data.frame(Scenario = "RegimeGARCH", Model = "FDA", Brier = cal_sim$fda$brier, ECE = cal_sim$fda$ece),
  data.frame(Scenario = "RegimeGARCH", Model = "TDA", Brier = cal_sim$tda$brier, ECE = cal_sim$tda$ece),
  data.frame(Scenario = "RegimeGARCH", Model = "BOTH", Brier = cal_sim$both$brier, ECE = cal_sim$both$ece),
  data.frame(Scenario = "RegimeGARCH", Model = "XGBoost", Brier = cal_sim$xgb$brier, ECE = cal_sim$xgb$ece))
write.csv(Table_Sim_Performance, file = file.path(TABDIR, "Table_Sim_Performance.csv"), row.names = FALSE)
write.csv(Table_Sim_AUC_Comparisons, file = file.path(TABDIR, "Table_Sim_AUC_Comparisons.csv"),
          row.names = FALSE)
write.csv(Table_Sim_Calibration, file = file.path(TABDIR, "Table_Sim_Calibration.csv"), row.names = FALSE)
g_roc_sim  <- plot_roc_four(sim_res)
g_cal_sim  <- plot_calibration(sim_res)
g_feat_sim <- plot_feature_by_class(sim_res)
ggplot2::ggsave(file.path(PLOTDIR, "Fig_SIM_ROC.png"), g_roc_sim, width = 7, height = 5, dpi = 300)
ggplot2::ggsave(file.path(PLOTDIR, "Fig_SIM_Calibration.png"), g_cal_sim, width = 7, height = 5, dpi = 300)
ggplot2::ggsave(file.path(PLOTDIR, "Fig_SIM_FeatureSummaries.png"), g_feat_sim, width = 8, height = 4,
                dpi = 300)
message("Simulation method finished. Tables in ", TABDIR, " | Plots in ", PLOTDIR)
```

Statements

Acknowledgements

The author would like to thank the anonymous referees and the handling editor for their careful reading and helpful comments, which have contributed to improving the article.

Author contributions

Conceptualization and formal analysis: C. Sözen; investigation: C. Sözen; methodology, original draft, review and editing: C. Sözen. All authors have read and agreed to the published version of the article.

Conflicts of interest

The author declares that there are no conflicts of interest or competing interests.

Data and code availability

The datasets generated and/or analyzed during the current study are available from the corresponding author on reasonable request and are also included in this article.

Declaration on the use of artificial intelligence (AI) technologies

The author used a generative AI tool only for language polishing.

Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

References

[1] Andersen, T.G., Bollerslev, T., Diebold, F.X., and Labys, P., 2003. Modeling and forecasting realized volatility. Econometrica, 71, 579–625. DOI: 10.1111/1468-0262.00418.

[2] Barndorff-Nielsen, O.E. and Shephard, N., 2002. Econometric analysis of realized volatility and its use in estimating stochastic volatility models. Journal of the Royal Statistical Society B, 64, 253–280. DOI: 10.1111/1467-9868.00336.

[3] Takens, F., 2006, October. Detecting strange attractors in turbulence. Proceedings of a symposium held at the University of Warwick 1979/80, pp. 366–381. Springer, Berlin, Germany. DOI: 10.1007/BFb0091924.

[4] Sauer, T., Yorke, J.A., and Casdagli, M., 1991. Embedology. Journal of Statistical Physics, 65, 579–616. DOI: 10.1007/BF01053745.

[5] Cohen-Steiner, D., Edelsbrunner, H., and Harer, J., 2005. Stability of persistence diagrams. Proceedings of the Twenty-First Annual Symposium on Computational Geometry. Association for Computing Machinery, New York, NY, USA, pp. 263–271. DOI: 10.1145/1064092.1064133.

[6] Edelsbrunner, H., and Harer, J., 2010. Computational Topology: An Introduction. American Mathematical Society, Providence, RI, USA. DOI: 10.1090/mbk/069.

[7] Breiman, L., 2001. Random forests. Machine Learning, 45, 5–32. DOI: 10.1023/A:1010933404324.
[8] Edelsbrunner, H., Letscher, D., and Zomorodian, A., 2002. Topological persistence and simplification. Discrete and Computational Geometry, 28, 511–533. DOI: 10.1007/s00454-002-2885-2.
[9] Zomorodian, A., and Carlsson, G., 2004. Computing persistent homology. Proceedings of the 20th Annual Symposium on Computational Geometry. Association for Computing Machinery, New York, NY, USA, pp. 347–356. DOI: 10.1145/997817.997870.
[10] DeLong, E.R., DeLong, D.M., and Clarke-Pearson, D.L., 1988. Comparing areas under correlated ROC curves. Biometrics, 44, 837–845. DOI: 10.2307/2531595.
[11] Coulibaly, B.D., Chaibi, G., and El Khomssi, M., 2024. Parameter estimation of the alpha-stable distribution and applications to financial data. Chilean Journal of Statistics, 15, 60–80. DOI: 10.32372/chJS.15-01-04.
[12] Bibi, A., 2024. Statistical inference for first-order periodic autoregressive conditional heteroskedasticity models. Chilean Journal of Statistics, 15, 169–191. DOI: 10.32372/ChJS.15-02-04.
[13] Horváth, L. and Kokoszka, P., 2012. Inference for Functional Data with Applications. Springer, New York, NY, USA. DOI: 10.1007/978-1-4614-3655-3.
[14] Hsing, T. and Eubank, R., 2015. Theoretical Foundations of Functional Data Analysis, with an Introduction to Linear Operators. Wiley, New York, NY, USA. DOI: 10.1002/9781118762547.
[15] Hyndman, R.J. and Shang, H.L., 2010. Rainbow plots, bagplots and boxplots for functional data. Journal of Computational and Graphical Statistics, 19, 29–45. DOI: 10.1198/jcgs.2009.08158.
[16] Shang, H.L., 2017. Functional time series forecasting with dynamic updating: An application to intraday particulate matter concentration. Econometrics and Statistics, 1, 184–200. DOI: 10.1016/j.ecosta.2016.08.004.
[17] Shang, H.L. and Kearney, F., 2022. Dynamic functional time-series forecasts of foreign exchange implied volatility surfaces. International Journal of Forecasting, 38, 1025–1049. DOI: 10.1016/j.ijforecast.2021.07.011.
[18] Fetitah, O., Attouch, M.K., Khardani, S., and Righi, A., 2021. Nonparametric relative error regression for functional time series data under random censorship. Chilean Journal of Statistics, 12, 145–170. URL: soche.cl/chjs/volumes/12/ChJS-12-02-02.pdf.
[19] Ortiz, S., and Becerra, O., 2024. On a Stahel–Donoho estimator with skewness-based random projection directions. Chilean Journal of Statistics, 15, 110–125. DOI: 10.32372/ChJS.15-02-01.
[20] Bauer, U., 2021. Ripser: Efficient computation of Vietoris–Rips persistence barcodes. Journal of Applied and Computational Topology, 5, 391–423. DOI: 10.1007/s41468-021-00071-5.
[21] Maria, C., Boissonnat, J.D., Glisse, M., and Yvinec, M., 2014. The GUDHI library: Simplicial complexes and persistent homology. Hong, H. and Yap, C.K. (eds.), Mathematical Software, Springer, Berlin, Germany, pp. 167–174. DOI: 10.1007/978-3-662-44199-2_28.
[22] Otter, N., Porter, M.A., Tillmann, U., Grindrod, P., and Harrington, H.A., 2017. A roadmap for the computation of persistent homology. EPJ Data Science, 6, 17. DOI: 10.1140/epjds/s13688-017-0109-5.
[23] Tauzin, G., Lupo, U., Tunstall, L., Burella Pérez, J., Caorsi, M., Medina-Mardones, A.M., Dassatti, A., and Hess, K., 2021. giotto-tda: A topological data analysis toolkit for machine learning and data exploration. Journal of Machine Learning Research, 22, 1–6. URL: www.jmlr.org/papers/v22/20-325.html.

[24] Bubenik, P., 2015. Statistical TDA using persistence landscapes. Journal of Machine Learning Research, 16, 77–102. URL: www.jmlr.org/papers/v16/bubenik15a.html.

[25] Adams, H., Emerson, T., Kirby, M., Neville, R., Peterson, C., Shipman, P., Chepushtanova, S., Hanson, E., Motta, F., and Ziegelmeier, L., 2017. Persistence images: A stable vector representation of persistent homology. Journal of Machine Learning Research, 18, 1–35. URL: www.jmlr.org/papers/v18/16-337.html.

[26] Reininghaus, J., Huber, S., Bauer, U., and Kwitt, R., 2015. A stable multi-scale kernel for topological machine learning. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. IEEE, Boston, MA, USA, pp. 4741–4748. DOI: 10.1109/CVPR.2015.7299106.

[27] Carrière, M., Cuturi, M., and Oudot, S., 2017. Sliced Wasserstein kernel for persistence diagrams. Proceeding of the International Conference on Machine Learning, pp. 664–673. URL: proceedings.mlr.press/v70/carriere17a.

[28] Kusano, G., Fukumizu, K., and Hiraoka, Y., 2018. Persistence weighted Gaussian kernels. Journal of Machine Learning Research, 18, 1–41. URL: jmlr.org/papers/v18/17-317.

[29] Gidea, M. and Katz, Y., 2018. TDA of financial time series: Landscapes of crashes. Physica A, 491, 820–834. DOI: 10.1016/j.physa.2017.09.028

[30] Guo, H., Xia, S., An, Q., Zhang, X., Sun, W., and Zhao, X., 2020. Empirical study of financial crises based on topological data analysis. Physica A, 558, 124956. DOI: 10.1016/j.physa.2020.124956.

[31] Fawcett, T., 2006. An introduction to ROC analysis. Pattern Recognition Letters, 27, 861–874. DOI: 10.1016/j.patrec.2005.10.010.

[32] Saito, T. and Rehmsmeier, M., 2015. PR plot is more informative than ROC for imbalance. PLoS ONE, 10, e0118432. DOI: 10.1371/journal.pone.0118432.

[33] Corsi, F., 2009. A simple approximate long-memory model of realized volatility. Journal of Financial Econometrics, 7, 174–196. DOI: 10.1093/jjfinec/nbp001.

[34] Yahoo Finance, 2025a. Intel Corporation (INTC) historical data. Retrieved 18 November 2025 from `https://finance.yahoo.com/quote/INTC/history`.

[35] Yahoo Finance, 2025b. ConocoPhillips (COP) historical data. Retrieved 18 November 2025 from `https://finance.yahoo.com/quote/COP/history`.

[36] Yahoo Finance, 2025c. NVIDIA Corporation (NVDA) historical data. Retrieved 18 November 2025 from `https://finance.yahoo.com/quote/NVDA/history`.

[37] Yahoo Finance, 2025d. The Walt Disney Company (DIS) historical data. Retrieved 18 November 2025 from `https://finance.yahoo.com/quote/DIS/history`.

[38] Chen, T. and Guestrin, C., 2016. XGBoost: A scalable tree boosting system. Proceedings of the International Conference on Knowledge Discovery and Data Mining. Association for Computing Machinery, San Francisco, CA, USA, pp. 785–794. DOI: 10.1145/2939672.2939785.

[39] Kitsos, C.P., Oliveira, A., and Nyamsi, U.E., 2025. Tail-adaptive generation of random numbers from a gamma-order normal distribution using the Ziggurat algorithm with a multivariate extension. Chilean Journal of Statistics, 16, 79–97. DOI: 10.32372/ChJS.16-01-05.