Feaws: A Hybrid Quantitative and Machine-Learning Framework for Cryptocurrency Price Forecasting and Tail-Risk Management

Karthikeyan NG

Independent Researcher

intrepidkarthi@gmail.com

February 2026

Abstract

This paper describes Feaws, an open-architecture forecasting engine for Bitcoin that I built over the course of roughly 4 months of evenings and weekends. The system stacks four layers—GARCH(1,1) volatility estimation, a gradient-boosted / Ridge regression ensemble for directional drift, a Merton jump-diffusion Monte Carlo simulator, and a five-state Hidden Markov regime classifier—into a single pipeline whose only forward-looking input is the current price. Every prediction is generated under a strict zero-lookahead walk-forward protocol: at each quarterly test date from Q1 2018 through Q1 2026, the model trains exclusively on data available up to that point, then forecasts 30- and 90-day price distributions. Over 30 out-of-sample 30-day windows the best configuration (ML + Factor) achieves 70.0 % directional accuracy and 96.7 % 90th-percentile confidence-interval coverage. A hard −20 % stop-loss circuit breaker caps portfolio drawdowns independent of model output. I report all eight model configurations, their scoring function, and the walk-forward backtest results without cherry-picking. The full codebase—including data pipelines, model training, and the Next.js dashboard—is available under an open-source licence.

Keywords: *Bitcoin, GARCH, Jump Diffusion, Gradient Boosting, Walk-Forward Validation, Tail Risk, Hidden Markov Model, Monte Carlo Simulation*

1 Introduction

I started working on Feaws in late 2025, not because I thought I could ”beat the market” in any absolute sense, but because the existing landscape of crypto prediction tools fell into two camps that both frustrated me. On one side, you had Twitter accounts and Telegram bots posting directional calls with no quantified uncertainty—just ”BTC to $150K by March.” On the other, the academic literature offered rigorous volatility models (Bollerslev, 1986; Engle, 1982) that were well-calibrated for traditional equities but rarely adapted to the 24/7, jump-heavy, regime-switching reality of cryptocurrency markets.

What I wanted was something in between: a system that gives a directional signal—up or down—but wraps it in honest probabilistic bounds. If the model says ”UP” with 62 % confidence, I want to also see that the 90th-percentile range is [$58,000, $84,000], so I know how uncertain that call really is. And I wanted every one of those numbers to come from a walk-forward backtest, not a single in-sample fit that looks brilliant on paper but falls apart the moment you deploy it.

Feaws is the result of that effort. It is a four-layer pipeline: (1) a GARCH(1,1) volatility engine with regime-conditional and EWMA components; (2) a drift engine that blends gradient-boosted regression, Ridge regression, a CAPM-style factor model, and regime-specific drift estimates; (3) a Merton jump-diffusion Monte Carlo engine that generates 5,000–10,000 price paths per forecast; and (4) a risk layer that computes VaR, CVaR, maximum drawdown distributions, and probability cones at the 50th, 68th, 90th, and 95th percentiles. The system ingests daily data for 28 assets—15 crypto tokens, 2 commodities (gold, silver), 5 tech stocks (NVDA, AAPL, MSFT, GOOGL, AMZN), and 6 global indices (S&P 500, NASDAQ, Nifty 50, Nikkei 225, Hang Seng, DAX)—sourced from Binance, yfinance, and CoinGecko APIs.

The contribution of this paper is not a novel algorithm but an engineering contribution: an end-to-end, reproducible pipeline that combines well-known quantitative-finance building blocks (GARCH, jump diffusion, factor models) with modern supervised learning in a way that respects the non-stationarity and fat tails of crypto returns. Everything described here is deployed and running; the live dashboard serves predictions updated on a daily cadence.

2 Related Work

Volatility modelling for financial time series dates to the seminal work of Engle (1982) on ARCH processes and Bollerslev (1986), who generalised these to the GARCH family. In the crypto domain, Katsiampa (2017) applied various GARCH specifications to Bitcoin returns and found that the AR-CGARCH model best captured the long-memory volatility component—a finding broadly consistent with the high persistence ($\alpha + \beta \approx 1.0$) we observe in our own GARCH(1,1) calibration.

Regime-switching models were introduced to econometrics by Hamilton (1989), who demonstrated that U.S. GDP growth could be decomposed into distinct expansion and contraction states via a Markov chain. Bouri et al. (2019) extended this to cryptocurrencies and showed that a two-regime model captures the bull-bear dynamics of Bitcoin returns better than single-regime alternatives. Feaws uses a five-state regime classifier (CRISIS, RISK_OFF, NEUTRAL, RISK_ON, EUPHORIA) motivated by the observation that crypto markets exhibit at least two distinct ”risk-on” and ”risk-off” states in addition to the classical two.

On the machine-learning side, Friedman (2001) introduced gradient boosting, which remains competitive with deep learning on tabular financial data (Grinsztajn et al., 2022). We pair a gradient-boosted regressor with a Ridge regression (Hoerl & Kennard, 1970) in a 60/40 blend to stabilise predictions against the overfitting that pure tree-based models are prone to when trained on fewer than 3,000 daily observations.

The Merton (1976) jump-diffusion extension to geometric Brownian motion is well-suited to crypto, where ±10 % single-day moves are not unusual. Our Monte Carlo engine calibrates jump intensity, mean, and standard deviation from 3-sigma tail events in the training window, producing fatter-tailed forecast distributions than a pure GBM would.

Finally, the idea that Bitcoin follows a roughly four-year halving cycle is widely discussed (PlanB, 2019, ”Stock-to-Flow” model), though the statistical evidence is debated. We include halving-phase features in the ML input set but let the gradient-boosted model learn their relevance rather than hard-coding a specific cycle thesis.

3 System Architecture

Feaws is organised into four composable layers. Each layer exposes a calibrate() method that trains exclusively on data available up to a cutoff date, enforcing the zero-lookahead constraint at the API level. The layers are:

3.1 Layer 1: Volatility Engine

The volatility engine provides three complementary estimates of conditional variance, all calibrated on log-returns $r_t = \ln(P_t / P_{t-1})$:

GARCH(1,1). We fit the standard specification $\sigma^2_t = \omega + \alpha \cdot \varepsilon^2_{t-1} + \beta \cdot \sigma^2_{t-1}$ using a Student-t innovation distribution and an AR(1) mean model. On the full training sample through February 2026, the estimated parameters are $\omega = 0.1448$, $\alpha = 0.0716$, $\beta = 0.9284$, giving a persistence $\alpha + \beta = 1.0000$. The near-unit persistence is expected: Bitcoin volatility clusters heavily, and shocks decay slowly. The unconditional annualised volatility is 69.1 %. For forward simulation we use the GARCH variance forecast (simulation method with 500 replications) to produce a time-varying daily volatility path $\sigma(t)$ for each of the next N days.

EWMA. As a cross-check, we compute an exponentially weighted moving average with λ = 0.94 (the Risk-Metrics convention). The resulting annualised volatility of 69.5 % is close to the GARCH estimate, which is reassuring—if the two diverged substantially it would suggest model misspecification.

Regime-conditional volatility. The HMM regime labels (Section 3.4) partition the return series into five states. We compute the annualised standard deviation within each partition. The results are intuitive: CRISIS = 126.1 %, RISK_OFF = 62.8 %, NEUTRAL = 68.4 %, RISK_ON = 64.8 %, EUPHORIA = 52.4 %. CRISIS volatility is roughly double that of the non-crisis regimes, consistent with the leverage effect in risk assets.

3.2 Layer 2: Drift Engine (ML + Factor Alpha)

The drift engine estimates the expected annualised return μ(t) that feeds into the Monte Carlo simulator. It has three sub-components:

ML signal.

A Gradient Boosting Regressor (GBR) with 200 trees, max depth 3, learning rate 0.03, 70 % subsample ratio, minimum 30 samples per leaf, and 70 % feature subsampling is paired with a Ridge regressor (α = 100) in a 60/40 weighted blend. The target variable is the realised percentage return over the forecast horizon (30 or 90 days), shifted back by the horizon length to avoid any leakage. A separate GBR classifier with identical hyperparameters predicts the binary direction (up/down) and provides a calibrated probability via predict_proba().

The input feature vector comprises 30 engineered features drawn from the 28-asset universe:

Feature Group	Features
BTC momentum	btc_ret_7d, btc_ret_14d, btc_ret_30d, btc_ret_60d, btc_ret_90d
BTC technicals	btc_rsi (14-day), btc_vol_30d, btc_vol_90d, btc_ma200_dist
Cross-asset returns	sp500_ret_14d, sp500_ret_60d, nasdaq_ret_14d, gold_ret_14d, gold_ret_60d, silver_ret_14d, nvda_ret_14d, nvda_ret_60d, nifty_ret_14d
Cross-asset ratios	gold_silver_ratio, eth_btc_ratio_14d
Rolling correlations	btc_nasdaq_corr_30d, btc_nasdaq_corr_90d, btc_gold_corr_30d, btc_gold_corr_90d
Regime & composite	regime (encoded −2..+2), composite_score
Halving cycle	halving_months_since, halving_phase, halving_cycle_drift, btc_ath_dist, btc_range_30d_pct

Feature importance (measured by Gini impurity in the GBR) as of February 2026 is dominated by btc_gold_corr_90d (23.8 %), halving_months_since (21.6 %), and halving_cycle_drift (10.4 %). The prominence of the gold correlation is notable—it reflects the post-2024 narrative shift in which Bitcoin is increasingly traded as a macro hedge alongside gold rather than purely as a risk-on tech proxy. I did not expect this feature to rank first when I originally included it; it was added almost as an afterthought because I happened to be pulling gold data for the factor model anyway.

Factor model.

A simple CAPM-style regression r_btc = α + β□·r_SP500 + β□·r_Gold + β□·r_NVDA + ε is fit on daily returns. The annualised intercept α captures the systematic excess return of Bitcoin relative to these three factors. This provides a structural, mean-reverting drift component that anchors the forecast when the ML signal is weak or noisy.

Regime drift.

The mean annualised log-return within each of the five HMM regimes is computed from the training window. In a CRISIS regime, for example, the historical drift is sharply negative, which pulls the blended μ(t) downward regardless of what the ML model says.

Signal blending.

The final drift is a weighted combination: μ_adj = w_ML · μ_ML + w_regime · μ_regime + w_factor · μ_factor, where the weights are defined per model configuration (Section 4). An optional drift cap clips μ_adj to [−cap, +cap] to prevent extreme annualised drift estimates from destabilising the Monte Carlo simulation.

3.3 Layer 3: Monte Carlo Engine (Merton Jump Diffusion)

Price paths are generated by the Merton (1976) jump-diffusion model:

$$dS / S = (\mu - \lambda\kappa)\, dt + \sigma(t)\, dW + J\, dN$$

where μ is the ML-adjusted drift, σ(t) is the GARCH time-varying volatility, W is a standard Wiener process, N is a Poisson process with intensity λ (average jumps per year), J ~ N(μ_J, σ_J) is the jump size, and κ = E[e^J − 1] is the jump compensator that ensures the drift is correctly adjusted for the expected jump contribution.

Jump parameters are calibrated from 3-sigma tail events in the training window. Over the full BTC history (2014–2026), we estimate roughly 5–8 jump events per year with a slightly negative mean jump size, reflecting the asymmetry of crypto crashes versus rallies.

Each forecast generates 5,000 paths (10,000 for the annual forecast). The terminal distribution at horizon T is summarised by percentiles (P5, P10, P25, P50, P75, P90, P95), from which we construct nested confidence intervals. A key design choice is alpha-risk separation: the Monte Carlo median is shifted to equal the ML point prediction, so the ML model determines where the distribution is centred (the alpha signal) while the stochastic simulation determines how wide it is (the risk model). This separation means the confidence intervals reflect genuine forecast uncertainty rather than model disagreement.

3.4 Regime Classification

A five-state Hidden Markov Model classifies the market into CRISIS, RISK_OFF, NEUTRAL, RISK_ON, and EUPHORIA. The states are determined by a composite scoring function that integrates BTC momentum (RSI, moving averages), cross-asset signals (S&P 500, VIX, gold), and on-chain metrics where available. The regime label at each test date is carried forward into the drift engine and the volatility engine, allowing both μ(t) and σ(t) to shift with the structural market state.

The regime-conditional volatilities reported in Section 3.1 validate the five-state decomposition: CRISIS periods exhibit roughly twice the volatility of EUPHORIA periods (126.1 % vs. 52.4 % annualised), suggesting the states capture genuinely different market microstructures rather than arbitrary partitions of the return distribution.

4 Model Configurations

Rather than committing to a single set of drift weights, Feaws evaluates eight distinct configurations that span the spectrum from pure ML reliance to pure historical drift. This is deliberate: I wanted to understand how much value the ML signal actually adds, and whether simpler configurations might be equally good or better. The configurations are:

Configuration	**w_ML**	**w_Regime**	**w_Factor**	**Drift Cap**	**Jumps**
Pure ML (Aggressive)	1.00	0.00	0.00	None	Yes
Pure ML (Conservative)	1.00	0.00	0.00	0.30	Yes
ML + Regime	0.60	0.40	0.00	None	Yes
ML + Factor	0.50	0.00	0.50	None	Yes
Balanced Hybrid	0.40	0.30	0.30	0.50	Yes
Regime Heavy	0.20	0.50	0.30	0.40	Yes
Contrarian (Fade ML)	−0.30	0.70	0.60	0.35	No
Historical Drift Only	0.00	0.00	0.00	None	Yes

The Contrarian configuration deliberately inverts the ML signal (w_ML = −0.30), betting that the model's directional call is more often a contrarian indicator. This is not as absurd as it sounds: in choppy, mean-reverting markets the ML model tends to chase momentum that quickly reverses. The Historical Drift Only configuration ignores all adaptive signals and uses the unconditional mean log-return—it serves as the naïve benchmark.

5 Walk-Forward Backtest Protocol

The backtest is the part of this project I am most careful about, because it is also the easiest part to get wrong. The protocol works as follows:

1. Test dates are set quarterly (January, April, July, October) for each year from 2018 through 2026—a total of up to 33 potential test points per configuration, though not all are evaluable (the most recent quarters lack a realised price at the forecast horizon).

2. At each test date, all engines (volatility, drift, Monte Carlo) are calibrated exclusively on data before that date. The training window grows monotonically: the January 2018 test uses ~4 years of BTC data; the January 2026 test uses ~12 years. No future information of any kind—prices, regime labels, factor returns—leaks into the training set.

3. Two horizons are evaluated: 30 days and 90 days. For each, the model generates a full price distribution (5,000 Monte Carlo paths), from which we extract the directional call, the ML-predicted price, and nested confidence intervals (50 %, 90 %, 95 %).

4. The actual price at the target date is compared against the prediction. Metrics collected: (a) directional accuracy (did we call up/down correctly?), (b) CI coverage (did the realised price fall within the predicted 90 % interval?), (c) median absolute percentage error.

5. Models are ranked by a composite score: 0.35 × dir_acc + 0.25 × CI90_cov + 0.20 × CI95_cov + 0.20 × max(0, 1 − avg_error). This weighting prioritises directional accuracy (the most actionable metric for

trading) while still rewarding well-calibrated uncertainty.

6 Empirical Results

6.1 30-Day Horizon

Configuration	N	Dir Acc %	CI90 %	CI95 %	Avg Err %	Score
ML + Factor	30	70.0	96.7	96.7	12.1	85.6
Balanced Hybrid	30	70.0	93.3	96.7	12.2	84.7
ML + Regime	30	66.7	96.7	96.7	13.4	84.1
Pure ML (Conservative)	30	66.7	93.3	96.7	12.5	83.5
Pure ML (Aggressive)	30	66.7	93.3	96.7	14.2	83.2
Historical Drift Only	30	63.3	93.3	96.7	13.3	82.2
Regime Heavy	30	56.7	93.3	100.0	13.2	80.5
Contrarian (Fade ML)	30	46.7	83.3	93.3	14.0	73.0

Several patterns stand out. First, every ML-based configuration outperforms the Historical Drift baseline on directional accuracy (63.3 %), confirming that the ML signal adds genuine value. The gap is not enormous—70.0 % vs. 63.3 %—but 6.7 percentage points of directional edge, compounded over dozens of quarterly decisions, translates to meaningful portfolio alpha.

Second, the Contrarian configuration performs worst on directional accuracy (46.7 %, below random chance), which tells us the ML signal is not a contrarian indicator over 30-day windows. This was a useful null check. At the 90-day horizon, the Contrarian does slightly better (51.7 %), suggesting that over longer horizons mean-reversion partially offsets the ML momentum signal.

Third, the confidence intervals are consistently well-calibrated. Six of eight configurations achieve $\geq$ 93.3 % coverage at the 90th percentile, meaning the realised price landed inside the predicted range at least 93 % of the time. The exception is the Contrarian (83.3 %), whose inverted drift shifts the distribution centre away from the realised outcome.

6.2 90-Day Horizon

Configuration	N	Dir Acc %	CI90 %	CI95 %	Avg Err %	Score
Pure ML (Conservative)	29	55.2	89.7	89.7	29.9	73.7
Balanced Hybrid	29	48.3	89.7	89.7	33.6	70.5

Configuration	N	Dir Acc %	CI90 %	CI95 %	Avg Err %	Score
Regime Heavy	29	48.3	86.2	89.7	33.1	69.8
Historical Drift Only	29	51.7	82.8	86.2	34.1	69.2
ML + Factor	29	48.3	82.8	86.2	41.0	66.6
Contrarian (Fade ML)	29	51.7	75.9	79.3	31.9	66.6
Pure ML (Aggressive)	29	55.2	72.4	82.8	55.2	62.9
ML + Regime	29	41.4	79.3	86.2	45.5	62.5

The 90-day results are humbling. Directional accuracy drops to the 48–55 % range for most configurations—barely better than a coin flip. This is not surprising: predicting the sign of a 3-month return in a market as volatile as Bitcoin is genuinely hard, and anyone claiming 70 %+ accuracy at this horizon should be viewed with suspicion. The ML + Factor configuration, which was the best at 30 days, drops to 48.3 % at 90 days—a good example of why horizon matters and why I report both.

The bright spot at 90 days is the confidence-interval coverage. Pure ML (Conservative) achieves 89.7 % at the 90th percentile, which is close to the nominal 90 %. The drift cap (0.30 annual) prevents extreme drift estimates from blowing the intervals too wide or too narrow. The take-away is that Feaws is better at quantifying uncertainty than at making point predictions—which, honestly, is the more useful property for risk management.

6.3 Feature Importance

Feature	Importance (%)
btc_gold_corr_90d	23.8
halving_months_since	21.6
halving_cycle_drift	10.4
btc_ath_dist	8.4
btc_ma200_dist	7.6
btc_vol_90d	3.6
btc_ret_60d	3.5
sp500_ret_60d	2.8
btc_ret_90d	2.5
nifty_ret_14d	2.4
btc_nasdaq_corr_90d	2.4
gold_ret_60d	2.2

The gold-BTC 90-day rolling correlation dominates at 23.8 %, followed by halving cycle features that together account for 32 % of total importance. This suggests the model has learned to condition its forecasts on where Bitcoin sits within the four-year halving cycle and how its relationship with gold is evolving—both of which are intuitively sensible macro drivers. BTC's distance from its all-time high (8.4 %) and from its 200-day moving average (7.6 %) round out the top five, acting as mean-reversion anchors.

What is notably absent from the top of the list is short-term BTC momentum (the 7- and 14-day return features). This surprised me early in development —I had assumed recent price action would be the strongest signal. But on reflection it makes sense: short-term momentum is noisy and mean-reverting in crypto, whereas the macro and cycle features are more persistent and therefore more predictable over 30-day horizons.

6.4 GARCH Parameters and Regime Volatilities

Parameter	Value
ω (omega)	0.1448
α (ARCH)	0.0716
β (GARCH)	0.9284
Persistence (α+β)	1.0000
Unconditional vol	69.1 %
EWMA vol (λ=0.94)	69.5 %

Regime	Annualised Volatility
CRISIS	126.1 %
RISK_OFF	62.8 %
NEUTRAL	68.4 %
RISK_ON	64.8 %
EUPHORIA	52.4 %

The GARCH persistence of exactly 1.0 (integrated GARCH, or IGARCH) means that volatility shocks are permanent in the model—a stylised feature that is commonly observed in crypto return series. The regime-conditional volatilities show that the five-state HMM captures meaningful heterogeneity: CRISIS periods are 2.4× more volatile than EUPHORIA periods. This dispersion justifies the computational cost of maintaining a regime-switching model rather than using a single unconditional volatility estimate.

7 Risk Management

No forecasting model, however well-calibrated, can protect a portfolio from a 50 % overnight crash in Bitcoin. The FTX collapse of November 2022, the Luna/UST de-peg of May 2022, and the March 2020 COVID crash all produced drawdowns that exceeded any reasonable confidence interval. For this reason, Feaws enforces a model-independent hard stop-loss: if the portfolio drawdown from peak value exceeds −20 %, all positions are liquidated regardless of the current forecast. This is implemented as a simple circuit breaker in the paper-trading engine.

The Monte Carlo engine also provides per-forecast risk metrics: Value-at-Risk (VaR) at the 1 % and 5 % levels, Conditional VaR (CVaR, i.e. expected shortfall), and the distribution of maximum drawdown across simulated paths. These metrics are displayed on the dashboard alongside the directional prediction, so a user can see not just ”the model says UP” but also ”the 5 % worst-case over this horizon is a 23 % loss.”

I want to be transparent about the limitations of this stop-loss design. A −20 % threshold is somewhat arbitrary; it was chosen to be wide enough to avoid false triggers during normal BTC volatility (the annualised vol of 69 % implies ±4.3 % daily moves are within one standard deviation) but tight enough to prevent catastrophic loss. In a true liquidity crisis where the market gaps down through −20 % without trading at intermediate prices, the actual realised loss could be worse. This is a known limitation of all stop-loss mechanisms in fragmented crypto markets.

8 Halving Cycle Integration

Bitcoin's supply issuance halves approximately every four years (November 2012, July 2016, May 2020, April 2024). The post-halving price behaviour, while not deterministic, shows a recurring four-phase pattern that Feaws encodes as engineered features:

Phase	Window	Empirical Drift
0 – Early Bull	0–6 months	+120 %/yr
1 – Main Bull	6–18 months	+80 %/yr
2 – Correction	18–30 months	−40 %/yr
3 – Accumulation	30–48 months	+30 %/yr

These drift estimates are measured across the 2016, 2020, and 2024 cycles. As of February 2026, we are approximately 22 months since the April 2024 halving—firmly in Phase 2 (Correction). The annualised forecast scenarios for the dashboard reflect this: the bear case uses a drift of −27.8 %/yr, the base case +43.3 %/yr (bottoming and early recovery), and the bull case +79.0 %/yr (the optimistic scenario where the correction has already ended and the next bull leg begins).

I want to emphasise that these cycle phases are not predictions—they are empirical regularities observed in a sample of three cycles, which is far too few to draw strong statistical conclusions. The model treats them as informative priors that the GBR can discount if the data tells a different story. In the feature importance results (Section 6.3), halving-related features account for 32 % of total importance, suggesting the model does find them useful, but this could equally reflect overfitting to a small-sample pattern. I flag this as a known caveat.

9 Implementation Details

The system is implemented in Python 3.11 (backtesting and data pipelines) and TypeScript/Next.js 15 (dashboard). The Python stack uses pandas, numpy, scikit-learn, the arch library for GARCH estimation, and scipy. The PostgreSQL database stores 165,202 daily candles across 28 symbols, sourced from Binance, yfinance, and CoinGecko APIs.

The refresh pipeline (scripts/refresh_signals.py) runs daily: it fetches the latest prices, re-computes features and regime labels, re-trains the ML models on the full available history, and writes a new forecast JSON. The dashboard reads this JSON server-side and renders predictions, backtest results, and the paper-trading portfolio in a single-page application. The total pipeline runtime is approximately 8 minutes on an M1 MacBook Pro.

A design choice worth noting: the walk-forward backtest in ensemble_models.py uses np.random.seed(42) before the main loop. This makes the Monte Carlo simulation deterministic, which is essential for reproducibility but means the specific numerical results reported here are conditional on that seed. I verified that changing the seed (e.g. to 0, 123, 999) shifts directional accuracy by at most ±2 percentage points and CI coverage by at most ±1 percentage point, indicating the results are robust to simulation randomness.

10 Limitations and Future Work

I want to be honest about what Feaws cannot do. The 90-day directional accuracy of 48–55 % is essentially a coin flip with a slight edge, and the 30-day accuracy of 70 %, while strong, is measured over only 30 out-of-sample windows—not enough to establish statistical significance at conventional levels (a binomial test of 21/30 correct against a null of 50 % yields $p \approx 0.04$, which is borderline). More data, both in terms of longer history and more granular test frequencies, would strengthen the claims.

The feature set is entirely price-based and macro-based. On-chain data (MVRV, SOPR, exchange inflows/outflows, mining hash rate) and sentiment data (social media volume, funding rates, options skew) are not yet integrated. These could plausibly improve the directional signal, particularly during regime transitions when price-based features lag.

The regime classifier uses a fixed five-state structure. A data-driven approach (e.g., fitting a Gaussian HMM with the number of states selected by BIC) might yield a more parsimonious decomposition. I chose five states based on qualitative domain knowledge rather than a systematic search, which introduces analyst bias.

Finally, the paper-trading module does not account for slippage, exchange fees, or execution latency. In live deployment, transaction costs would erode returns, particularly for configurations that trade frequently. A future version should incorporate a realistic execution model.

11 Conclusion

Feaws demonstrates that a stack of well-understood quantitative-finance components—GARCH volatility, gradient-boosted regression, factor models, and Merton jump diffusion—can produce Bitcoin forecasts that are directionally useful (70 % at 30 days) and probabilistically well-calibrated (96.7 % CI90 coverage). The system does not claim to predict the future with certainty; its value lies in wrapping every forecast in honest, backtested uncertainty bounds.

Building this system taught me—more than any textbook could—that the difficult part of quantitative finance is not the maths but the discipline: resisting the temptation to peek at future data during backtesting, reporting configurations that performed badly alongside those that performed well, and acknowledging that a 30-observation backtest is informative but not conclusive. I hope the open-source release of Feaws contributes to a culture of transparency in crypto forecasting, where predictions come with error bars and backtest protocols are reproducible.

References

Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. Journal of Econometrics, 31(3), 307–327.

Bouri, E., Shahzad, S. J. H., Roubaud, D., Kristoufek, L., & Lucey, B. (2019). Bitcoin, gold, and commodities as safe havens for stocks: New insight through wavelet analysis. The Quarterly Review of Economics and Finance, 77, 156–164.

Engle, R. F. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. Econometrica, 50(4), 987–1007.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. Annals of Statistics, 29(5), 1189–1232.

Grinsztajn, L., Oyallon, E., & Varoquaux, G. (2022). Why do tree-based models still outperform deep learning on tabular data? Advances in Neural Information Processing Systems, 35, 507–520.

Hamilton, J. D. (1989). A new approach to the economic analysis of nonstationary time series and the business cycle. Econometrica, 57(2), 357–384.

Hoerl, A. E., & Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. Technometrics, 12(1), 55–67.

Katsiampa, P. (2017). Volatility estimation for Bitcoin: A comparison of GARCH models. Economics Letters, 158, 3–6.

Merton, R. C. (1976). Option pricing when underlying stock returns are discontinuous. Journal of Financial Economics, 3(1–2), 125–144.

Appendix A: Model Scoring Function

Each model configuration is scored per horizon using:

$$S = 0.35 \times dir\_acc + 0.25 \times CI90\_cov + 0.20 \times CI95\_cov + 0.20 \times \max(0, 1 - avg\_err)$$

where dir_acc is the fraction of test windows where the predicted direction matched the realised direction, CI90_cov and CI95_cov are the fractions where the realised price fell within the 90th and 95th percentile intervals, and avg_err is the mean absolute percentage error of the point prediction divided by 100 (so an average error of 12 % contributes 0.88 to the last term). The weights were chosen to prioritise directional accuracy (the most decision-relevant metric) while penalising poor calibration.

Appendix B: Reproducibility

The complete source code is available at https://github.com/intrepidkarthi/feaws (to be made public upon acceptance). To reproduce the results:

1. Clone the repository and install dependencies: pip install -r requirements.txt

2. Set up a local PostgreSQL database named ”feaws” and run scripts/seed_historical.py to populate price history.

3. Execute python3 backtesting/ensemble_models.py to run the full walk-forward backtest. Output is written to backtesting/results/ensemble_dashboard.json.

4. Start the dashboard with cd dashboard && npm run dev to view results interactively.

All random seeds are fixed (np.random.seed(42)) for deterministic Monte Carlo output. Python 3.11, scikit-learn 1.3+, arch 6.0+, and numpy 1.24+ are required.