**Requirements Document — Leak-Proof ML Trading Signals on Minute-Bar Crypto**

**0) Audience & Purpose**

This document specifies exactly what to build, how to build it, and why—so an engineer or an AI assistant can implement a **deployable** trading signal generator for crypto (minute bars) that **avoids overfitting** and **generalizes**. It covers the end-to-end pipeline: data → features → labels → validation → models → calibration → ensembling → trade construction → risk → backtesting → deployment → monitoring → retraining. It leaves no room for ambiguity and encodes guardrails against common finance ML errors (leakage, non-stationarity, backtest mirages).

**Hard constraints:**

* **No data leakage** (strict causality at every step).
* **All metrics after fees/slippage** only.
* **Validation must be purged & embargoed, walk-forward, and nested for hyperparameters.**
* **Signals must be calibrated probabilities and mapped to expected value (EV), with hysteresis and risk limits.**

**1) Problem Context & Goals**

**1.1 Context**

* Data: 1-minute OHLCV + trades for a single crypto asset (extendable to many). High noise, regime shifts, heavy tails, non-stationarity.
* Current use: classification to produce **Long / Short / Neutral** signals; probabilities used as confidence.

**1.2 Primary Goal**

Produce **clear, tradable entry/exit signals** with a **repeatable statistical edge** that persists out-of-sample/live, measured **after realistic costs**.

**1.3 Secondary Goals**

* Robustness across regimes; low sensitivity to hyperparameters.
* Controlled turnover and drawdowns.
* Transparent, auditable pipeline with reproducible research and deployment.

**2) Definitions & Notation**

* tt: bar index; all features at tt use information **available at or before** bar close tt.
* Price PtP\_t: use **close** unless specified.
* Log return: rt=ln⁡Pt−ln⁡Pt−1r\_t = \ln P\_t - \ln P\_{t-1}.
* Rolling volatility: e.g., σt\sigma\_t from a causal rolling window (std of rr).
* **Triple-barrier labeling**: upper barrier +kσ+k\sigma, lower barrier −kσ-k\sigma, and max horizon HH minutes; label = which barrier hits first (or 0 if neither).
* **Purge**: drop training samples overlapping in time with validation label horizons.
* **Embargo**: hold-out gap after each validation fold to prevent leakage via close-by outcomes.
* **Uniqueness / concurrency weights**: down-weight overlapping events.
* **EV**: expected value per trade net of costs.

**3) System Architecture (High-Level)**

1. **Ingestion & Integrity**
2. **Bar Construction** (time bars + info-driven bars)
3. **Feature Engineering** (strictly causal)
4. **Labeling** (triple-barrier + sample weights)
5. **Split & Validation** (purged, embargoed, nested walk-forward)
6. **Model Training** (regularized baselines)
7. **Probability Calibration**
8. **Ensembling** (across models & horizons)
9. **Signal Gating & Execution Mapping** (EV thresholds, hysteresis, cool-downs)
10. **Risk Controls & Sizing** (vol target, limits, kill-switches)
11. **Backtesting Engine** (after-cost, event-driven)
12. **Deployment** (real-time inference)
13. **Monitoring & Retraining** (drift, calibration, stability)

**4) Data Requirements**

**4.1 Inputs (must-have per bar)**

* Timestamp (UTC), Open, High, Low, Close, Volume.
* Trades count (if available) and notional volume (quote & base).
* **Fees** (maker/taker schedule), **estimated spread** series (from best bid/ask if available; else proxy).
* Optional: microstructure (order book top-of-book), but not required for v1.

**4.2 Timezone & Ordering**

* Store raw in UTC. Convert for displays as needed. Bar close at **end of minute**. Ensure strict monotonic timestamps; fill missing minutes explicitly with NaNs then forward engineering **must not** peek into future.

**4.3 Data Integrity Checks (blocking)**

* No duplicate timestamps.
* No negative prices/volumes.
* Gaps identified and logged.
* Roll-forward/backfill **disallowed** for features—use only causal windows.

**5) Bar Construction**

**5.1 Time Bars**

* Primary: 1m bars (original).
* Aggregates: 5m, 15m, 60m computed **causally** (downsample by grouping past minutes).

**5.2 Information-Driven Bars (optional v1, required v1.1)**

* **Volume bars**: emit a bar each time cumulative base volume ≥ threshold V∗V^\*.
* **Dollar bars**: emit a bar each time cumulative quote notional ≥ threshold D∗D^\*.
* Thresholds selected to roughly match the typical data density of 1–5m bars.
* Maintain separate datasets per bar type/horizon; **do not merge** into a single interleaved sequence. Models ensemble across them later.

**6) Feature Engineering (Causal Only)**

**6.1 Core Features (Minimal, Robust)**

* Returns: rt,1r\_{t,1} (1m), rt,5r\_{t,5}, rt,15r\_{t,15} (compounded).
* Volatility: rolling std of 1m returns over windows W∈{15,60,240}W \in \{15, 60, 240\}.
* Ranges: HL%, CO%, true range proxies (causal).
* Volume/trade intensity: rolling z-scores (causal).
* Skew/kurtosis: rolling higher moments on returns (small windows; clamp outliers).
* **Fractional differencing**: price series fractionally differenced with d∈[0.2,0.6]d \in [0.2, 0.6] tuned to first achieve stationarity on **training only**. Include as features; **do not replace** returns.

**6.2 Denoising**

* **Allowed**: **causal** EMA/IIR smoothing on features; **no zero-phase** filters (e.g., filtfilt) and **no future look-ahead** (wavelets only if implemented strictly causal; otherwise skip).
* If using Butterworth/IIR: apply single-pass causal form; document order & cutoff; verify no phase compensation from the future.

**6.3 Scaling & Leakage Control**

* Feature scaling (if used) must be **fit on training folds only** and applied to validation/test.
* Rolling statistics for each t must use data ≤t\leq t.
* Drop any feature that requires future bars.

**6.4 Redundancy Pruning**

* Remove features with ∣ρ∣>0.95|\rho| > 0.95 within training folds.
* Stability selection: keep features with consistent importance/sign across folds.

**7) Labeling — Triple-Barrier Events**

**7.1 Parameters**

* H∈{30,60,90}H \in \{30, 60, 90\} minutes (grid).
* k∈{1.0,1.5,2.0}k \in \{1.0, 1.5, 2.0\} multiple of rolling σt\sigma\_t (choose σ\sigma window from {60, 240} mins).
* Profit barrier: +k⋅σt+k \cdot \sigma\_t; stop barrier: −k⋅σt-k \cdot \sigma\_t.
* Starting price: close at t (or mid if available; be consistent across pipeline).

**7.2 Rules**

* Simulate forward from t up to t+Ht+H (causal only) to detect first touch of either barrier.
* Label yt∈{+1,−1,0}y\_t \in \{+1, -1, 0\}: +1 if upper hits first; -1 if lower; 0 if neither.
* **Overlap accounting**: compute **event concurrency** (how many active events overlap at each time) and derive **uniqueness weights** wt∝1/concurrencytw\_t \propto 1/\text{concurrency}\_t; normalize to mean 1 on the training set.

**7.3 Class Imbalance**

* Use class weights or focal loss; **no oversampling**. Preserve label base rates.

**8) Validation Protocol — Purged, Embargoed, Nested Walk-Forward**

**8.1 Fold Construction**

* Use K=5K=5 chronological folds. For each fold:
  + **Train** on past window.
  + **Validate** on next segment.
  + **Purge**: remove from training any samples whose event horizon overlaps validation.
  + **Embargo**: skip an additional gap of length HH after validation start.

**8.2 Nested Tuning**

* Inner loop: choose hyperparameters by maximizing **after-cost Sharpe** on the inner validation (same purge/embargo).
* Outer loop: report performance distribution across folds (Sharpe, Sortino, turnover, max DD).
* **Final test**: the most recent untouched window, reserved from the start; used for paper-trading only. No tuning based on this set.

**9) Models & Hyperparameters**

**9.1 Baseline 1 — Logistic Regression (Elastic-Net)**

* Inputs: selected features.
* Solver: liblinear/saga.
* Grid: C∈{0.01,0.1,1,10}C \in \{0.01, 0.1, 1, 10\}, l1\_ratio ∈{0,0.25,0.5,0.75,1}\in \{0, 0.25, 0.5, 0.75, 1\}.
* Class weights: “balanced” or inverse frequency.
* Use **sample weights wtw\_t** from uniqueness.

**9.2 Baseline 2 — LightGBM**

* Objective: binary or multiclass (for {-1,0,+1}); recommend **binary one-vs-rest** with meta-labeling later; for v1, binary “act vs no-act” is acceptable if chosen.
* Key constraints (regularization first):
  + max\_depth ∈ {3,4,5}
  + num\_leaves consistent with depth (≤ 2^(depth+1))
  + min\_data\_in\_leaf ∈ {200, 500, 1000} (tune vs dataset size)
  + feature\_fraction ∈ {0.6, 0.8}
  + bagging\_fraction ∈ {0.6, 0.8}, bagging\_freq > 0
  + lambda\_l1 ∈ {0, 1e-3, 1e-2, 1e-1}
  + lambda\_l2 ∈ {0, 1e-3, 1e-2, 1e-1}
  + learning\_rate ∈ {0.02, 0.05, 0.1}; early\_stopping\_rounds=200; num\_boost\_round up to 5000.
* Use **sample weights** wtw\_t.
* Monotonic constraints: only if domain knowledge supports (optional).

**9.3 Baseline 3 — CatBoost**

* Ordered boosting **on**.
* Depth ∈ {3,4,5}; learning\_rate ∈ {0.02,0.05,0.1}; l2\_leaf\_reg ∈ {1,3,10,30}; bagging\_temperature ∈ {0,1}; subsample ∈ {0.6,0.8} if not using ordered bootstrap.
* Early stopping: 200 rounds.
* Sample weights: wtw\_t.

**Note:** Deep models (TCN/LSTM/Transformers) are **out of scope for v1**; revisit only if v1 is stable and survives robustness tests.

**10) Probability Calibration**

* Method: **Platt scaling** (logistic) or **isotonic regression**.
* Fit calibrator **per fold** on the **validation** predictions only; apply to corresponding test segment.
* Evaluate **Brier score** and reliability curves.
* Store calibrated ptp\_t for execution mapping.

**11) Ensembling**

**11.1 Across Models**

* Weighted average of calibrated probabilities from **Logit**, **LightGBM**, **CatBoost**.
* Weights learned **only on training** via a constrained optimizer (non-negative, sum to 1) to maximize **after-cost Sharpe** on validation; no access to test.
* Simpler fallback: equal weights.

**11.2 Across Horizons / Bar Types**

* Maintain separate model streams for 1m, 5m, 15m time bars and (when available) dollar/volume bars.
* Ensemble by averaging probabilities with horizon-specific weights (constrained as above).
* Optional rule: **regime gating** (e.g., high vol → favor shorter horizons).

**12) Signal Gating & Execution Mapping**

**12.1 EV Calculation**

For a long signal with TP = +θ+\theta (in bps), SL = −θ-\theta, calibrated probability pp of profit barrier hit within HH, and round-trip cost CC (bps):

EV=p⋅θ−(1−p)⋅θ−C=(2p−1)θ−CEV = p \cdot \theta - (1-p)\cdot \theta - C = (2p-1)\theta - C

* Enter only if **EV > 0** **and** **EV > M** (safety margin MM, e.g., 2× median cost over last week).

**12.2 Thresholding, Hysteresis, Cool-Downs**

* Two thresholds Tenter>TexitT\_{enter} > T\_{exit} on pp or EV to reduce churn (hysteresis).
* Require persistence: signal must satisfy threshold for **N consecutive bars** (N∈{2,3}).
* After exit: **cool-down** of CdC\_d minutes (e.g., 10–30) before re-entry in same direction.

**12.3 Position Sizing & Volatility Target**

* Raw size s\\*=clip(2p−1,0,1)s^\\* = \text{clip}(2p-1, 0, 1).
* Scale to meet **volatility target**: choose leverage so that rolling 1-day realized vol of PnL ≈ target ν\nu (e.g., 10% annualized).
* Hard caps: max leverage LmaxL\_{max}; max position QmaxQ\_{max}; max daily turnover UmaxU\_{max}.

**12.4 Risk Controls & Kill-Switches**

* Per-trade stopout at SL; per-day stop after max realized drawdown DdayD\_{day}.
* Rolling **max drawdown** limit over last 90 days triggers de-risk (halve size) and at next threshold triggers **all-stop** until manual review.
* Capacity checks: reject trades if expected slippage > threshold.

**13) Costs & Slippage Model (Backtest & Live)**

* **Fees**: maker/taker per exchange schedule; encode explicitly.
* **Spread**: use best quotes if available; else rolling proxy from high-low micro-ranges.
* **Slippage**: function of volatility × size; baseline: slip=α⋅σ1m⋅size/ADV\text{slip} = \alpha \cdot \sigma\_{1m} \cdot \sqrt{\text{size}/\text{ADV}}. Tune α\alpha conservatively.
* **Partial fills**: simulate with queue priority = taker unless resting limit order logic is implemented (not in v1).
* Apply costs at **execution time** (signal → order → fill) in backtest.

**14) Backtesting Engine (Deterministic, Event-Driven)**

* **Inputs**: bars, signals, costs, sizing rules.
* **Latency assumption**: orders placed at bar close, filled at **next bar open** price ± half-spread ± slippage (configurable).
* **Order types**: market only in v1; limit logic out of scope.
* **Position accounting**: FIFO; PnL includes fees and slippage.
* **Outputs**: per-trade log, per-bar PnL, equity curve.

**15) Evaluation Metrics (After Costs Only)**

* Annualized **Sharpe** and **Sortino**.
* **Hit rate**, **average win/loss**, **expectancy**.
* **Turnover** (daily, annualized).
* **Max drawdown**, **Calmar/MAR**.
* **Stability**: std of fold Sharpe; inter-quartile range across folds.
* **Calibration**: Brier score, reliability curves.
* **Capacity**: slippage vs size sensitivity.

**Model selection** optimizes for **after-cost Sharpe** subject to constraints: turnover ≤ target, max DD ≤ cap, calibration error ≤ threshold.

**16) Robustness & Stress Testing**

* **Block/bootstrap**: stationary/block bootstrap of returns to test persistence.
* **Cost stress**: ±(2–3)× fees/spread/slippage.
* **Parameter knockouts**: perturb key HPs ±20% to test brittleness.
* **Regime slicing**: high vs low volatility, trend vs range periods.
* **p-value style checks**: reality-check style multiple testing control (at least report selection count and adjusted expectations).

Pass/fail criteria: strategy maintains positive after-cost Sharpe and acceptable drawdowns in at least **80%** of stress scenarios.

**17) Deployment Requirements**

**17.1 Real-Time Inference**

* Inference cycle: every minute at bar close + 1–2s buffer.
* Max end-to-end latency (feature calc → signal → order): **< 3 seconds**.
* Deterministic feature state managed via a simple **feature store** (rolling windows updated incrementally).

**17.2 Packaging & Reproducibility**

* Fixed random seeds per fold and model.
* Config-driven (YAML): data paths, HP grids, thresholds, costs.
* Version everything: code, configs, datasets (hash/snapshot).
* Experiment tracker (any): log HPs, metrics, artifacts (models, calibrators).

**17.3 Monitoring**

* Live metrics: rolling Sharpe, turnover, hit rate, slippage vs model, calibration drift (Brier), latency.
* Alerts: breach of kill-switch thresholds, missing data, latency > SLA, calibration drift > tolerance.
* Logging: per-decision record (features hash, model version, probs, EV, action, fill).

**17.4 Retraining**

* Cadence: weekly (or after **N** new days or regime change trigger).
* Process: freeze last live model; retrain via the exact pipeline; validate; paper-trade 1–2 weeks before promotion.
* Rollback: ability to revert to last stable model instantly.

**18) Implementation Checklist (Top-Down)**

1. **Data ingestion & integrity checks** done.
2. **Bar builders** (1m, 5m, 15m; + volume/dollar in v1.1) ready.
3. **Causal features** implemented; leakage tests passed.
4. **Triple-barrier labels** + **uniqueness weights** implemented.
5. **Purged+embargoed nested walk-forward** splitter implemented and unit-tested.
6. **Baselines** (Logit, LGBM, CatBoost) train with **sample weights** and early stopping.
7. **Calibration** per fold (Platt/Isotonic) with stored calibrators.
8. **Ensemble** weights fitted on training folds only; applied out-of-fold.
9. **Execution mapping** (EV thresholds, hysteresis, cool-downs) implemented.
10. **Vol target sizing**, **risk caps**, **kill-switches** wired.
11. **Backtester** simulates orders with costs and outputs trade logs + metrics.
12. **Reports**: cross-fold summaries, robustness suite, final untouched paper-trade report.
13. **Deployment** scaffold: inference loop, monitoring, alerting.
14. **Runbook** for retraining & rollback.

**19) Acceptance Criteria**

* **Leakage tests**: unit tests that deliberately try to use future info must fail; all rolling ops verified causal.
* **Validation protocol**: code proves purge/embargo lengths equal to label horizon HH.
* **Metrics**: on cross-fold OOS, after-cost Sharpe > 0 with stable dispersion; calibration Brier < naive baseline; turnover within configured limit.
* **Stress suite**: positive after-cost Sharpe in ≥ 80% scenarios; drawdowns under cap.
* **Paper-trade window** (most recent period): after-cost Sharpe within **50–100%** of cross-fold median; turnover and DD consistent.
* **Monitoring**: live dashboards and alerts configured; dry-run shows signals/alerts firing as expected.
* **Reproducibility**: re-running with same seeds reproduces metrics within statistical jitter.

**20) Micro-Goals by Phase (Why each exists)**

* **Labels & Splits**: Remove noise and leakage → trustworthy edges.
* **Features**: Small, causal, stationarized set → lower variance.
* **Models**: Regularized baselines → strong generalization.
* **Calibration**: Turn scores into **actionable probabilities** for EV mapping.
* **Ensembles**: Reduce variance across models/horizons.
* **Execution & Risk**: Convert edge to PnL while controlling churn and drawdowns.
* **Backtest Realism**: Prevent cost/latency fantasies.
* **Monitoring & Retraining**: Catch drift and keep the edge alive.

**21) Guardrails & Non-Negotiables**

* **Never** use non-causal filters (filtfilt, centered windows, future-aware transforms).
* **Never** tune on the final test/paper-trade window.
* **Always** evaluate **after costs**; report costs separately.
* **Document** every assumption (latency, slippage, fees, horizons).

**22) Deliverables**

1. **Code package** with modules: ingestion, bars, features, labels, cv\_splitter, models, calibration, ensemble, execution, risk, backtester, reports, deploy.
2. **Config files** (YAML) with all parameters and grids.
3. **Unit tests** (features causality, purge/embargo, calibration separation).
4. **Integration tests** (end-to-end backtest on a small dataset).
5. **Experiment logs** and **artifacts** (models + calibrators per fold).
6. **Backtest report** (cross-fold + stress + paper-trade).
7. **Runbooks** (deployment, monitoring, retraining, rollback).

**23) Optional Extensions (Post-v1)**

* **Meta-labeling**: a secondary classifier deciding when to act on primary signals.
* **Uncertainty gating** via **conformal prediction** or CV dispersion thresholds.
* **Regime models**: state detection to select horizon weights.
* **Info-driven bars** (volume/dollar) moved from optional to default after evaluation.
* **Deep models** (TCN) under same validation strictness; keep only if they beat baselines **after costs** and survive robustness tests.
* **RL**: only with a high-fidelity simulator and strict OOS controls.

**24) Appendix — Pseudocode Outlines**

**24.1 Triple-Barrier Labeling**

for each index t:

start\_price = Close[t]

upper = start\_price \* exp(+k \* sigma\_t)

lower = start\_price \* exp(-k \* sigma\_t)

for tau in 1..H:

p = Close[t+tau] # or mid, strictly future-only within label calc

if p >= upper: y[t] = +1; T\_end[t] = t+tau; break

if p <= lower: y[t] = -1; T\_end[t] = t+tau; break

if no barrier hit by H: y[t] = 0; T\_end[t] = t+H

Compute concurrency over all active [t, T\_end[t]]; uniqueness weight wtw\_t inversely proportional to concurrency, normalized.

**24.2 Purged, Embargoed Walk-Forward**

split timeline into K folds chronologically

for fold i:

val = segment\_i

embargo\_gap = H

train = data before val, excluding:

- any t whose [t, T\_end[t]] overlaps val

- embargo\_gap minutes after val start

fit inner-CV for HPs on train

evaluate on val

**24.3 EV Thresholding & Sizing**

p = calibrated\_prob\_long

theta = k \* sigma\_t\_in\_bps

EV = (2\*p - 1) \* theta - Cost

if EV > max(0, M) and persistence\_condition\_met and not in\_cooldown:

size\_raw = clip(2\*p - 1, 0, 1)

size = vol\_target(size\_raw)

place\_order(size)

else:

hold\_or\_exit()

**25) Why We Integrated Each Approach**

* **Triple-barrier labeling**: focuses learning on **meaningful moves**, reducing label noise from 1-min chops.
* **Purged & embargoed CV**: eliminates leakage from overlapping horizons; realistic OOS estimates.
* **Downsampling & info-driven bars**: improve signal-to-noise; we **add** them as parallel views to ensemble, not as a single replacement.
* **Causal smoothing & fractional differencing**: stabilize features while preserving memory and avoiding leakage.
* **Regularized baselines (Logit/LGBM/CatBoost)**: strong on noisy tabular data with built-in variance control.
* **Probability calibration**: converts raw scores into **trustworthy probabilities** for EV mapping and sizing.
* **Ensembling across models/horizons**: reduces variance and regime sensitivity.
* **EV gating, hysteresis, cool-downs, vol-targeting**: translates statistical edge into **tradable** performance with controlled churn and risk.
* **Robustness & stress tests**: filter out backtest mirages and parameter accidents before capital is at risk.

**End of spec.**  
If you need this converted into a project scaffold (folder structure, config templates, and test stubs), say the word and I’ll lay it out.