The right way to make this “smart” is to (1) precompute fast daily outcomes (your cube), and (2) attach **context features** about “what the world looked like at decision time,” then (3) learn a mapping from context → best combo + expected return + confidence. Below is a concrete blueprint you can implement on GCP with minimal extra moving parts.

**1) What to measure (context features)**

These are computed **as-of the open of the decision session** (RTH/AH) for each symbol.

**BTC context (market driver)**

* Weekend gap % (Fri close → Mon pre-open), last N hours/days return, 5/20/60-min momentum
* Volatility regime: realized vol (e.g., 24h, 7d), ATR-style range, VVIX proxy via BTC IV (if available) or realized σ buckets (Low/Med/High)
* Drawdown state: distance from 20/50/200-day highs, z-score of BTC return last 24–72h
* Macro direction flags: “BTC up ≥3% over weekend” (boolean), same for −3%

**Stock context (each miner)**

* Ratio regime: z-score of (BTC / stock) vs your N-day baseline (what you already compute)
* Mean-reversion pressure: ratio z-score slope over the last 60–180 minutes
* Liquidity/quality: dollar volume percentile today vs past 30 days, spread proxy (if you have it)
* Idiosyncratic move: stock’s residual vs a BTC-beta regression over last week (how much it’s deviating from BTC)
* Corporate/event flags: “news/merge/earnings window” (manual override or simple calendar)

**Session/context**

* Session type (RTH/AH/ALL), Day-of-Week, “post-holiday Monday,” month-end/quarter-end flags
* Overnight BTC move bucket: [≤−3%, −3 to −1, −1 to +1, +1 to +3, ≥+3]
* Volatility bucket (tri-state) for BTC and for the ratio

These are all **cheap** to compute from data you already ingest.

**2) What to precompute nightly (keeps reports instant)**

Keep your **Daily Outcome Cube** (one row per symbol, date, method, buy\_pct, sell\_pct, window) with day\_return and n\_trades. Then add a **Feature Snapshot** for the same symbol,date,window that captures the “as-of” context features above. This lets you train models that answer: *“Given today’s context, which (method, buy, sell) is likely best?”*

**Tables (BigQuery recommended):**

* minute\_join (source of truth; you already have this pattern)
* day\_features (your daily baselines by method)
* roi\_daily\_cube (partition by et\_date, cluster by symbol, method) → precomputed day\_return
* context\_snapshot\_daily (partition by et\_date, cluster by symbol) → all features as of session start

Minimal context\_snapshot\_daily columns:

et\_date DATE, symbol STRING, window STRING, lookback\_n INT64,

btc\_ret\_wend\_gap NUMERIC, btc\_ret\_24h NUMERIC, btc\_vol\_7d NUMERIC,

ratio\_z NUMERIC, ratio\_z\_slope NUMERIC, stock\_dollar\_vol\_pct NUMERIC,

dofw INT64, is\_post\_holiday BOOL, vol\_bucket STRING, gap\_bucket STRING, ...

**3) How to label and learn (so it “picks” settings today)**

**3.1 Labels**

Build several labels from the cube to match different decision goals:

* **Argmax label (“best combo”):** For each (symbol,date,window), compute the **best** (method,buy,sell) by day\_return. That’s your **classification** target.
* **Top-K returns:** Keep the actual day\_return for all combos to train **regressors** that predict expected return (or quantiles).
* **Hit label:** day\_return > 0 (or > fee/slippage) for **binary** models (probability of being profitable).

**3.2 Models (start simple, upgrade later)**

* **Stage 1 (simple & robust):**
  + **Classification** (XGBoost/LightGBM) to predict the best **method** given context.
  + **Two regressors** to predict E[day\_return | context, method] for **buy%** and **sell%** on a grid, then pick argmax.
  + Or a single regressor that takes (context, method, buy, sell) and scores any combo; at runtime you score the ~900 combos quickly in memory and choose the top.
* **Stage 2 (uncertainty-aware):**
  + **Quantile regression** (e.g., 0.2/0.5/0.8) to get a distribution of returns.
  + **Bayesian last-N smoothing** to stabilize recent performance by regime.
* **Stage 3 (online):**
  + **Contextual bandit** (e.g., LinUCB/NeuralUCB/Thompson) to continuously explore/exploit combos with guardrails. Keep this optional; your environment has costs/slippage, so be conservative.

**Walk-forward training:**  
Train on months 1–9, validate on month 10, test on month 11; slide forward. Use **time-series CV**, never shuffle across time.

**4) Confidence score (actionable, not hand-wavy)**

Design it so it **down-weights flimsy patterns** and **up-weights stable regimes**. Combine **probability of outperformance**, **uncertainty**, and **sample support**:

Let:

* = predicted mean day return for the chosen combo (net of fees/slippage)
* = predicted 20th/80th percentile returns (uncertainty band)
* = predicted probability that return > 0 (from a classifier or from the distribution)
* = effective sample size for *similar contexts* (e.g., BTC gap bucket + vol bucket + DoW) in your history window
* = **Bayesian-smoothed** success rate with a Beta prior (α=3, β=3 is a good start)

**Confidence, 0–1:**

* is a support scale (e.g., 100).
* tunes how much you punish wide prediction intervals.

**Guardrails:**

* **Floor/Cap** confidence (e.g., 0.05–0.95).
* **News override:** if has\_major\_news → cap confidence at 0.3 unless manually lifted.

**5) Budget allocation using confidence**

Translate confidence to dollars **safely**:

* **Scaled Kelly (conservative):**  
  Estimate “edge” and loss probability . Approximate payoff ratio from your trade logs (avg win / avg loss).

with to stay conservative and cap (e.g., 10% per symbol). Multiply by **confidence**.

* **Confidence-weighted risk parity:**  
  Weight each symbol (σ = recent symbol volatility), then cap per-symbol and per-sector.

**6) Reports & dashboards (what you’ll actually look at)**

1. **Regime Playbook (Mon 9:30 ET)**
   * Inputs: BTC weekend gap bucket, vol bucket → show top 10 combos by predicted return/confidence per symbol.
   * Include **budget suggestions** and expected ranges (quantiles).
   * “Reason why” panel: top features (feature importance/SHAP) driving the pick.
2. **Live Tuner**
   * Every 5–10 minutes recompute context (BTC move, ratio z-score slope) and re-rank combos.
   * Show confidence trend, any **degradation alerts** (concept drift).
3. **Combo Leaderboard (historical explorer)**
   * Slice by BTC gap bucket, vol bucket, DoW.
   * Heatmaps of (buy%, sell%) → expected return and hit-rate.
4. **Confidence & Coverage**
   * For each symbol, track **effective sample size** in the current regime and decay curves (so you know when the model is guessing).
5. **Risk/Overvaluation Monitor**
   * Ratio z-score vs historical band; when > +2σ, **auto reduce budget** (unless news override set).
   * “Mean-reversion probability next X minutes” as a quick dial.
6. **Attribution & Post-Mortem**
   * For each trading day, log **context**, **recommended combo**, **actual chosen combo**, **confidence**, **budget**, and **outcome**.
   * This becomes labeled data for online improvements.

**7) How to build it on GCP (wiring)**

* **BigQuery** for roi\_daily\_cube + context\_snapshot\_daily (partitioned by date, clustered as noted).
* **Cloud Run (Python)** service:
  + Endpoint /recommend:
    1. Pull current context (fast SQL on minute\_join + last N metrics).
    2. Score combos (or score the grid via a regressor) → choose top K.
    3. Compute confidence and budget weights.
    4. Return JSON (top settings per symbol + confidence + budget).
  + Endpoint /learn: log outcomes for the post-mortem table.
* **Model training**: weekly job (Cloud Run Job) reading BQ tables, training XGBoost/LightGBM; store the model in Cloud Storage; load on startup.
* **Caching**: Memorystore (Redis) for recent contexts and last recommendations (helps if many users hit at once).
* **Manual overrides**: tiny overrides table in BQ/Cloud SQL (symbol, start\_ts, end\_ts, reason, cap\_confidence, force\_budget).

**8) Quick start checklist**

* Extend your nightly job to create context\_snapshot\_daily.
* Keep roi\_daily\_cube (your 45k rows/day) for instant aggregation.
* Build a **training view** joining context\_snapshot\_daily ⨝ roi\_daily\_cube.
* Train a first-pass regressor on (context, method, buy, sell) → day\_return.
* Implement the confidence formula above and a conservative budget rule.
* Ship **Regime Playbook** and **Live Tuner** dashboards first; iterate.

If you want, I can draft:

* The BigQuery DDL for context\_snapshot\_daily
* A Python/Cloud Run scoring handler that ingests current context, scores all 900 combos per symbol in ~milliseconds, and outputs (settings, confidence, budget) with simple JSON you can drop into Firebase.