ChatGPT

Signal Generation Code Analysis (nothing14yyx/quant_trade)

1. Technical Indicators & Parameters

The signal generation relies on a rich set of technical indicators across multiple timeframes (15m, 1h, 4h, d1). Key indicators include:

- **Relative Strength Index (RSI)** – 14-period (standard length) for each timeframe [1]. This is a typical choice and generally reasonable for capturing momentum on hourly/daily scales.
- **Moving Average Convergence Divergence (MACD)** – Standard (12, 26, 9) periods for fast EMA, slow EMA, and signal line [2]. MACD histogram values (`macd_hist`) are used as features, which is a common trend/momentum gauge.
- **Bollinger Bands** – 20-period with 2.0 standard deviation width [3]. They use the **Bollinger Band %B** (`boll_perc`) to measure price position within bands and **band width** (`bb_width`) to gauge volatility. A 20-period window and 2σ is a classic setting, typically reasonable for mid-term volatility assessment.
- **Stochastic Oscillator** – %K period 14 with smoothing 3 (so 14,3,3 stochastic) [4]. This captures short-term overbought/oversold conditions; 14,3,3 is a standard configuration.
- **Average True Range (ATR)** – 14-period ATR is calculated and normalized as ATR% of price [1], a volatility measure. Period 14 is standard and suitable for a roughly two-week window on daily data or half-day on hourly data.
- **Average Directional Index (ADX)** – 14-period to indicate trend strength [5] [6]. ADX 14 is a typical default; they also compute **ADX delta** (day-to-day change) to see trend strengthening/weakening. These parameters are reasonable; ADX⩾25 is later used as a threshold for trending market.
- **Commodity Channel Index (CCI)** – 14-period [6]. CCI 14 is slightly shorter than the classic 20, but still within normal usage.
- **Money Flow Index (MFI)** – 14-period (custom-calculated) [7] [8]. Standard period; it's incorporated as both MFI value and money flow ratio.
- **Moving Averages** – Short EMA of 10 and long EMA of 50 to create an **EMA difference** feature [9] [10], plus SMA5, SMA10, SMA20 for trends and cross-ratios [10]. These are typical short/medium-term lengths (5/10/20 for SMAs, 10/50 for EMAs) and serve to gauge trend direction (e.g. price vs. moving averages).
- **Ichimoku Cloud** – Tenkan-sen 9, Kijun-sen 26 (default) used to derive conversion/base lines and cloud thickness [11] [12]. These defaults are standard for Ichimoku; features like conversion–base difference and cloud thickness inform momentum and support/resistance.
- **Donchian Channels** – 20-period highs/lows [13] [14] for breakout detection (`donchian_perc` and range width). 20 is a common choice (roughly monthly on daily).
- **Keltner Channels** – 20-period with ATR(2) band offsets [15] [16]. Standard configuration to measure volatility-based envelope (similar period to Bollinger).
- **On-Balance Volume (OBV)** – running sum to capture volume trend, with OBV delta (first difference) [17]. No period (cumulative indicator), so no parameter issue here.
- **Volume Metrics** – Volume Rate of Change (5-period) [18], short vs long volume MA ratio (volume vs 10-period SMA, and vs 30-period mean) [19], and a `vol_breakout` flag feature. These help identify

unusual volume surges. The chosen lengths (5, 10, 30) are short-term oriented, which should effectively detect volume spikes or dry-ups.

- **Others**: Pivot points (daily classic pivots R1/S1) [20], support/resistance levels and breakout flags (20-period window for local min/max) [21] [22], **Williams %R** (implicitly via `willr` features), **Streaks** of bullish/bearish candles, **shadows/wicks** ratios, and even external metrics (fear & greed index, funding rates etc.) are used. Most of these use standard windows (often 20 or 14).

Overall, the **indicator parameters are mostly standard defaults**, which are reasonable starting points. For example, RSI-14, MACD(12,26,9), Bollinger(20,2.0) are well-known baseline settings [1] [3]. These should generally be fine. One potential consideration is whether certain timeframes might benefit from slight tuning – e.g. **very short-term signals** (like the 15m confirmation) might respond faster with shorter RSI or stochastic periods. Currently the code uses the same default periods even on lower timeframes (RSI_15m also length 14). This uniformity keeps things consistent, but if signals are sluggish, experimenting with a shorter window on the 15m or 1h indicators (e.g. RSI-7 or 9) could make them more responsive. Similarly, the **Supertrend** uses a 7-period ATR and 3.0 multiplier [23], which is quite sensitive (short ATR window). While this can catch quick trend flips, it may generate noise – a slightly longer period (e.g. 10 or 14) might reduce whipsaws if that has been an issue. Overall, though, there's no glaringly "unreasonable" parameter – they align with common technical analysis practice.

2. Signal Logic and Oversold/Overbought Handling

The signal generation logic is quite sophisticated, involving an ensemble of factor scores and an AI model prediction. However, parts of it may be **overly mechanical**, relying on hard thresholds that could blunt its adaptability in extreme conditions. Specifically:

- **Mechanistic Thresholds for Extremes:** The code explicitly avoids taking positions in extreme **oversold/overbought conditions**. For example, after computing a tentative direction, it cancels any short signal if daily RSI<30 or daily CCI<-100 (oversold), and cancels any long if RSI>70 or CCI>100 (overbought) [24]. This is a safety mechanism to prevent selling bottoms or buying tops. While prudent, it can be too rigid: in a strong downtrend, RSI can stay <30 for a long time – the system would refuse to short, potentially missing profitable trend-continuation trades. Likewise, in a roaring uptrend with RSI >70, it won't go long (flagging "overbought"), possibly exiting too early or entirely missing the rally. In essence, the logic **prevents acting in the direction of an extreme** but does **not actively capitalize on the potential reversal** except by staying flat. This could indeed mean the strategy doesn't properly catch "oversold bounces" – it avoids further shorting when oversold, but it doesn't explicitly signal a long entry to play the rebound. The result is a flat/no-position during what might be a prime recovery opportunity.

- **Attempt to Handle Extremes via Score Adjustments:** The system does include some nuanced handling for extremes in its scoring. For instance, if the daily context is extreme (RSI < 25 or > 75, CCI out of ±100), it **deweights the daily AI model's influence by 30%** [25]. This means the model's vote is toned down during extreme fear or euphoria, presumably to allow other factors (or reversal logic) to have more say. Additionally, if price breaks a daily support level and daily RSI is <30 (a likely oversold crash), the code switches the market regime to "range" and sets a `rev_dir = 1` flag indicating a possible upward reversal [26]. These are smart touches – effectively saying "this breakdown might be a false breakdown due to oversold conditions." However, these adjustments might not be sufficient to generate a counter-trend signal; they mostly prevent the system from being overly bearish in a crash scenario (or overly bullish in a blow-off top). In other words, the system is **better at avoiding bad trades in extremes than at actively taking advantage of the reversal**. This cautious approach can lead to signals that are a bit behind the curve on V-shaped reversals.

- **Potential Over-Mechanical Behavior:** The signal logic uses a voting system and multiple small rules, which, though logically sound, could make the behavior somewhat mechanical. For example, they require a certain number of factors to align before taking a trade (a "min vote" of 2 and confidence threshold 0.12 by default) [27] [28]. If everything isn't just right, the system stands aside. This conservatism helps avoid false signals but may also cause the strategy to sit out real opportunities if they don't check all the boxes. Similarly, the code counts how many timeframes' trend signals agree with the trade; if fewer than one timeframe aligns, it cancels the trade direction [29]. This means pure contrarian or single-timeframe setups won't trigger – the trade needs at least one of (1h,4h,d1) trending in that direction. In practice, that filters out some noise, but it also means the very beginning of a new trend (where all higher timeframes are still opposite) won't be taken. The logic is essentially "prove to me at least on one timeframe that this trade is with trend; otherwise no trade." It avoids many traps but at the cost of missing some early entries. This mechanical alignment rule may delay entries until the move is already underway (reducing profit potential from catching inflection points).

In summary, the signal logic is robust but **possibly too rigid around extreme conditions**. It excels at not getting caught in obviously bad trades (e.g. shorting an oversold market) [24], but in doing so it might fail to capitalize on the very moment when the market turns (since it doesn't flip to long either at that point). The code could benefit from a more dynamic oversold/overbought strategy – for instance, allowing **small exploratory positions on reversal signals** (divergences, candlestick patterns, etc.), or using gradually tightening stops rather than a hard "no trade" at RSI 70+. Right now, a strong trend that stays overbought will yield zero long signals from the system, which might be overly cautious. A more nuanced approach (e.g. if ADX is high, allow continuation trades even if RSI overbought, perhaps with trailing stop protection) could improve performance in runaway trends.

3. Filtering Sideways/Choppy Markets

Choppy, non-trending markets are notoriously difficult. The code does attempt to recognize and adapt to low-trend conditions, though there might be room to strengthen this area:

- **ADX-Based Regime Filter:** They compute an average of 1h, 4h, d1 ADX and classify the market as "trend" if the average ADX $\geqslant$ 25, else "range" [30]. This is a simple but effective criterion – ADX below ~20–25 typically indicates a sideways or weak-trend market. By using 25 as the cutoff [31], the system flags sideways conditions and can respond accordingly. In fact, it does respond: the position sizing logic uses a smaller base position coefficient for range vs trend (default 0.40 vs 0.60) [32] [33]. So in a range regime, even if a signal fires, the default position size is scaled down to ~2/3 of what it would be in a trending regime. This is a form of filtering – it doesn't outright ban trades, but it **de-emphasizes trades during sideways markets**.

- **Volume and Volatility Filters:** The code also checks for low volatility/volume conditions. Notably, if the market is in "range" regime **and** the 1h volume is very low (volume vs its 30-bar average < 0.2, by default) **and** the fused signal score is barely above the base threshold (within 0.02) **and** there's no consensus among factors, then it halves the position size again [34]. This essentially says: "if we're in a likely no-trend, low-volume lull and the signal isn't strong, cut the position further (and possibly to zero if it falls below min position)". Indeed, if position size after this falls below a minimum (e.g. <5%), they drop the trade entirely [35]. This mechanism acts as a **filter for dull, indecisive market periods**, preventing the system from getting chopped up. It's a smart way to avoid many false signals in sideways markets – by requiring either some volume spike or stronger conviction to take a trade when conditions are listless.

- **Trend Confirmation Requirements:** As mentioned, the system needs at least one timeframe's trend factor to agree with the trade direction [36]. This too serves as a filter against trading purely range-bound countertrend blips. E.g., in a flat market, trend signals on all timeframes might be near zero; without any positive alignment, the system won't trade on a tiny oscillation.

- **Volume Breakout Factor:** On the flip side, the presence of a `vol_breakout` feature (which is 1 if a significant volume surge is detected on 1h/d1) means the model encourages trades during high-volume breakouts. In the factor computation, a volume breakout adds positively to the trend score [37], and in the voting system it's given a small weight (1 out of the vote total) [38]. This indicates the system tries to participate when volume confirms a potential move, which is usually when a range might be turning into a trend. If no volume spike, the vol_breakout vote is 0 and won't boost the trade signal. So indirectly, lack of volume can keep the vote below the required threshold (since min_vote=2) and thus filter out trades in quiet times.

Overall, **yes, the system has mechanisms to handle sideways markets**. The ADX regime classification and volume-based downsizing are in place and seem quite reasonable. One thing it doesn't do is a **hard "do not trade" toggle for ranges** – it still may trade during ranges, but with reduced size and only on stronger signals. If further filtering is needed (if performance in sideways conditions is still an issue), possible improvements could include:

- Incorporating a **volatility filter**: e.g., require ATR% or Bollinger band width to exceed a minimum before taking trend-following trades. Currently, ATR% is used in dynamic thresholding, but one could simply not trade if volatility is extremely low (to avoid whipsaws when price is stuck in a tight channel). The code already indirectly covers this (since in ultra-low volatility, fused score likely remains below threshold), but an explicit check could be added for clarity.
- Using **additional trend confirmation**: e.g., price above/below a certain MA for longs/shorts. They do include moving average cross info (via `ema_diff` and `sma_5_20`), but perhaps requiring, say, 1H price above the 4H 10-SMA for long (and vice versa) could filter out range noise. This might be redundant given their factor design, however.
- **Avoiding low timeframe chop**: maybe enforce that if 15m confirmation is weak or contradictory, don't trade. They already use a 15m `confirm_15m` score as part of the environment/risk check. Ensuring that e.g. `confirm_15m` is not negative for a long trade (or magnitude above some small value) could help avoid very short-term counter swings.

In summary, the system is **already designed to de-prioritize trades in non-trending phases** through ADX and volume awareness. If needed, these filters could be tightened (for example, raise the ADX threshold to 30 to be pickier, or flat-out skip trades if ADX < 15 indicating ultra-flat). Introducing a direct "no-trade regime" state for extreme low volatility could be considered, but with the risk of missing the breakouts out of the range. The current approach of scaling down and requiring extra confirmation is a balanced solution.

4. Other Potential Issues Affecting Signal Accuracy

Beyond indicator choices and filtering logic, a few other factors could impact signal accuracy and real-world reliability:

- **Signal Lag:** The use of multi-timeframe indicators means some signals inherently lag. For example, daily features (RSI_d1, CCI_d1, etc.) only update once per day at the daily close. In intraday trading, this introduces a slight lag in reflecting new trends – the system is effectively trading today based partly on yesterday's daily indicator readings. If the market's character changed dramatically since yesterday (say a sharp reversal this morning), the daily RSI won't

show it until tomorrow. The code does merge 1h data with the most recent 4h and d1 data using backward-asof alignment [39] (so each 1h bar is tagged with the last closed 4h and daily values), which is correct to avoid lookahead. But it means **intraday signals rely on stale higher-timeframe context** until those higher bars close. The impact is the model may react slower to regime changes – e.g., if a big breakout occurs, 1H indicators will respond immediately, but the system might wait for 4H or daily confirmation to really scale up (since until then, ADX_d1 or trend_d1 factors might still be "cold"). This conservatism can reduce false positives (good for avoiding noise), but it does **delay some valid signals** – a trade-off between accuracy and timeliness.

- **Data & Execution Delays:** In live trading, any delay in data feed or processing could cause signals to come a bit late. The strategy uses order book imbalance, funding rates, and external metrics (like Fear & Greed index) that might not update every bar. They handle this by forward-filling and smoothing [40], but traders should ensure these inputs are timely. For example, if the funding rate or an external index is significantly delayed, it might skew the signal. There's also multi-threading in use (with locks to protect data) – thread scheduling delays shouldn't be big, but it's worth noting that thread complexity can sometimes cause subtle timing issues (the code appears careful with locks though). Ensuring that **all indicators are computed on the latest available prices** and that the generate_signal function is called promptly when a new bar completes will help minimize any latency in signals.

- **Backtest Logic & Bias:** The repository includes a parameter optimization routine (using Optuna and grid search) [41] to tune weights and thresholds. If not done carefully, this process can lead to **overfitting** – the system may have been tweaked to perform extremely well on historical data, at the risk of not generalizing. The use of time-series cross-validation in `param_search.py` suggests they attempted to mitigate this (they split the data in time to test on forward segments). Still, the sheer number of adjustable knobs (various factor weights, thresholds like `base_th`, `gamma`, etc., as well as feature selections) presents a risk of **model complexity**. High complexity can reduce real-time signal accuracy if market conditions shift away from the historical patterns the model fit. In other words, the more parameters tuned to past data, the more the live performance might suffer if those patterns change. It's important to validate the strategy on truly out-of-sample periods and monitor if any particular filter or weight is consistently hurting live performance (which might indicate an overfit aspect).

- **Signal Instability and Conflicts:** With so many components (AI model vote, momentum factors, mean-reversion factors, volume factors, etc.), there is a possibility of **signal conflict** or flip-flopping. The code accounts for this to some extent (e.g. a `conflict_filter` that sets position_size to 0 if conflicting signals trigger [42]). However, rapid oscillation of signals (especially around the threshold boundary) can occur if the fused score hovers near zero. The dynamic thresholding (`base_th` adjusting with volatility and recent performance) helps by requiring a clear margin before flipping direction, and there's also a cooldown concept (not explicitly seen in snippet, but implied by `_cooldown` usage) to prevent immediate reversal after an exit. Despite these, users should watch for cases where the strategy enters and exits too frequently in a short time – that would indicate whipsaw noise is getting through. Fine-tuning the `confidence_vote` threshold or adding a slight hysteresis (e.g., once a signal triggers, require a larger opposite signal to flip) could further stabilize this if needed.

- **Stop Loss / Take Profit placement:** The code calculates take-profit and stop-loss based on ATR and also adjusts them if model-predicted rise/drawdown is available [43] [44]. One must ensure these stops are actually being used and managed correctly in live trading (the output includes them). If stops are too tight (ATR multiplier defaults 1.5 TP / 1.0 SL) the system might get stopped

out often, impacting the **realized accuracy vs. signaled direction**. Conversely, too wide stops can reduce risk-adjusted returns. The values seem reasonable, but it's a point of consideration: the signal may be accurate directionally, but if the risk management around it is suboptimal, the strategy could still lose money despite "being right." Reviewing the chosen multipliers and possibly making them dynamic (e.g. tighter in ranging markets, wider in trending) might improve real-world reliability.

In summary, the main non-obvious issues that might reduce signal accuracy are **signal lag from multi-timeframe confirmation requirements**, potential **overfitting from heavy parameter tuning**, and the general complexity which could lead to occasional conflicting signals or suboptimal stop placements. The backtest logic appears solid (no lookahead, realistic fee/slippage assumptions, etc.), but ongoing evaluation is needed to ensure the model's complexity doesn't secretly encode some bias that won't hold in future data.

Issues Summary

Combining the points above, the following key issues or observations were identified in the signal code:

- **Standard Indicator Settings** – The strategy uses mostly default indicator parameters (RSI-14, MACD 12-26-9, Bollinger 20-2, etc.) [1] [3]. These are reasonable, but there is little customization per timeframe. Fine-tuning certain lengths (especially on the lowest timeframe used for confirmation) could potentially improve responsiveness. Also, the Supertrend length is quite short (7) [23], which might cause whipsaws. This is more of a minor calibration issue than a bug – overall the choices are sane.

- **Overbought/Oversold Handling** – The logic avoids trading at extreme indicator values: it cancels longs if RSI>70 (overbought) and shorts if RSI<30 (oversold) [24]. This protects against buying tops/selling bottoms, but it also means the system exits or stays out just when a trend is strong or a reversal could yield profit. There's no mechanism to **positively trade the reversal** (only to avoid trading in the prior direction), so the strategy may miss quick rebound opportunities. Additionally, in prolonged trends that stay in overbought/oversold territory, this rule could keep the system on the sidelines, reducing profit potential from trend-following.

- **Potentially Mechanical Rules** – Requirements like a minimum number of confirming votes, trend alignment across timeframes, etc., while logically sound, might make the system too strict or slow. It may filter out many false signals (good for accuracy), but at the cost of sometimes filtering out early legitimate signals (bad for opportunity). The net effect on accuracy needs to be monitored – the signals it does take are likely high-quality, but if it skips many profitable ones, that's an issue for performance (though not "accuracy" in the sense of being wrong, rather a conservative bias).

- **Sideways Market Filters** – The code does detect and adapt to low-trend conditions via ADX and volume measures [30] [34]. However, it doesn't completely avoid trading in ranges; it just reduces position size. If range-bound whipsaws still prove costly, stronger filters could be needed (like not trading at all when volatility is below a threshold, or using longer confirmation). Currently, the mechanisms in place (reduced size, higher threshold for signal in ranges) seem reasonable, but there's no dedicated feature like "don't trade if price stuck in Bollinger band < X width for Y time" which some systems use. If the strategy still chops in ranges, that's an area for improvement.

- **Signal Lag and Responsiveness** – By incorporating 4H and daily indicators, signals might lag fast market moves. The strategy waits for bar closes and often for multi-bar confirmation. This can reduce losing trades (improves accuracy rate) but means entries are often late and exits might be late too. In fast markets, the accuracy of prediction might suffer simply because the data feeding the decision is slightly old. There's a trade-off here between accuracy and timeliness.

- **Data and Execution Latency** – Not a code bug per se, but a practical concern: using external data (funding rates, etc.) introduces points of failure or delay. Any latency in computing features or placing orders could widen the gap between signal decision and execution price, effectively lowering accuracy vs. ideal conditions. The code uses thread locks and likely runs in a live loop; ensuring that it processes new data as soon as available will help keep signals aligned with current market state.

- **Complexity & Overfitting** – The strategy's complexity (ensemble of factors, AI model, dynamic thresholds) is a double-edged sword. It can adapt to many scenarios, but it also means many parameters were fitted. If some are overfit (e.g. the precise weight values or threshold of 0.05 in delta boosts), the real-time signal accuracy could degrade when market regime changes. The backtest/optimization procedure [41] must have been thorough; even so, continuous retraining or periodic re-optimization may be necessary to maintain accuracy as market conditions evolve.

- **Signal Quality vs. Quantity** – The combination of all filters likely yields high-accuracy signals when they do appear, but with fewer signals. If accuracy is defined as win-rate, this is fine; but if the user meant capturing more correct moves, the system might be missing some (preferring not to trade rather than trade without full confirmation). This is a design choice, but it's worth noting as an "issue" in the sense that the user might improve overall returns by slightly loosening filters to catch more moves – albeit with the risk of lowering per-trade accuracy.

Suggested Code Modifications Prompt

To address the above issues and improve signal accuracy and reliability, here is a consolidated prompt outlining recommended code modifications:

Modify the trading signal generation logic as follows (aimed at increasing accuracy and real-world reliability):

1. **Enhance Oversold/Overbought Logic:** Instead of flatly zeroing signals when RSI/CCI hit extreme values [24], implement a more nuanced approach. Allow partial or conditional positions in strong trends despite extreme oscillators, and actively detect reversal patterns: for example, if RSI < 30 but starting to rise and other momentum indicators turn positive, allow a small contrarian long ("oversold bounce") signal. Similarly, in overbought uptrends, consider keeping a reduced long position or using a trailing stop rather than exiting entirely at RSI > 70. This will enable the system to **capture rebounds** and not exit trends prematurely, while still respecting extreme conditions. In code, this could mean removing or relaxing the hard `direction=0` setting on RSI/CCI thresholds and instead incorporating those conditions into the scoring or position sizing (e.g. reduce position by X% when overbought, rather than dropping to zero).

2. **Dynamic Trend-Continuation Filter:** Refine the extreme-condition handling by factoring in trend strength (ADX). For instance, permit trades even with high RSI if ADX is above a high threshold (meaning a very strong trend) – the rationale being that in super-trending markets, overbought

can persist. Code-wise, wrap the RSI/CCI cancel logic in a condition that checks trend regime: only cancel for oversold/overbought if the broader trend is weak or range (`regime != "trend"` or ADX below, say, 30). This prevents the system from cutting off winners in a powerful trend.

3. **Improve Sideways Market Avoidance:** Introduce a stricter filter to avoid choppy markets. For example, if ADX on all timeframes is very low (avg ADX well below 25) and volatility metrics like `bb_width` or ATR% are at extreme lows, skip new trade entries entirely. This could be a simple if-condition in `generate_signal`: if `regime=="range"` and `atr_pct_1h` (or `bb_width_1h`) is below a small percentile, return signal 0. Another approach: require the **vol_breakout** flag to be true (or momentum to exceed a threshold) to initiate a position in range regime. This ensures the strategy waits for a convincing breakout from consolidation, thereby **filtering out whipsaws**. Adjust the `signal_filters.min_vote` or `confidence_vote` higher during identified range periods to demand stronger consensus before trading.

4. **Increase Short-Term Responsiveness:** To reduce signal lag, incorporate faster indicator feedback on the lower timeframe. For instance, add a **shorter-period RSI** (like 7-period) or a momentum indicator on the 15m data and include it in the confirmation. This could catch abrupt moves sooner. In code, compute `rsi_fast_15m` or similar and use its extreme values or crosses as an additional factor for immediate term signals. Also, consider using the current (live) value of certain indicators rather than waiting for bar close in critical situations – e.g., if price moves far beyond a support/resistance intra-bar, you might trigger a signal mid-bar (this is more complex and may require infrastructure for intra-bar data, so as a simpler step, using the 15m data as you already do is likely sufficient). Essentially, **tweak the feature set or timing to get quicker confirmation** for entries, so the system isn't always one bar late on fast reversals.

5. **Reduce Overfitting & Simplify**: Revisit the plethora of parameters and consider simplifying or making some values more generic to avoid overfitting to past data. For example, if certain delta-boost thresholds or minor factor weights (those 0.03, 0.05 increments in `DELTA_PARAMS` and factor formulas) [45] aren't significantly impacting decisions, remove or neutralize them. Aim for a more parsimonious model that relies on the most robust signals (price/volume action and a few key indicators) rather than many small adjustments. In code, this could involve reducing the number of factors in the vote (maybe drop the weakest contributing factor), or setting some of the learned parameters (like `gamma`, `rev_boost`) to fixed reasonable values instead of highly-tuned ones. This will make the strategy more **robust to regime changes** and easier to maintain.

6. **Adaptive Stop Loss/Take Profit:** To improve real-world reliability (which affects the accuracy of profitable vs. losing trades), adjust the TP/SL strategy. Use market regime info to scale ATR multipliers – e.g., in range markets use tighter stops (1.0 ATR or even less) and in trending markets use wider stops (1.5–2 ATR) to allow winners to run. The code can dynamically choose `tp_mult`/`sl_mult` based on `regime` or recent volatility. Also, ensure that the stop-loss is never too tight relative to the entry conditions (perhaps enforce a minimum stop distance of, say, 0.5% even if ATR is tiny). These tweaks will prevent premature stop-outs in noise and improve the win rate of signals.

7. **Continuous Evaluation & Learning:** Implement a mechanism to periodically evaluate signal success and adjust thresholds automatically. For example, track the rolling accuracy of each factor's signals (the code computes factor IC scores offline; this could be extended online). If a particular factor or filter isn't contributing to accuracy (say the "short_mom" vote often conflicts with actual outcomes), consider reducing its weight over time. This could be a semi-automatic calibration that runs weekly to tweak `vote_weights` in the config. While not a direct

code change to signal logic, it’ s a meta-improvement to ensure the strategy stays accurate as market dynamics evolve.

By making these changes, the strategy should become **more responsive to profitable opportunities** (notably oversold bounces and breakouts) while still avoiding the worst trades in ranges or exhausted trends. It will maintain a high accuracy rate but also capitalize on more situations, and be better adapted to live market conditions. Each modification should be tested carefully to ensure it indeed boosts performance without introducing undue risk or false signals.

---

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 40 helper.py
https://github.com/nothing14yyx/quant_trade/blob/97952698bdebd012f7baef14230bf69822c2b006/quant_trade/utils/helper.py

24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 42 43 44 45 robust_signal_generator.py
https://github.com/nothing14yyx/quant_trade/blob/97952698bdebd012f7baef14230bf69822c2b006/quant_trade/robust_signal_generator.py

39 feature_engineering.py
https://github.com/nothing14yyx/quant_trade/blob/97952698bdebd012f7baef14230bf69822c2b006/quant_trade/feature_engineering.py

41 param_search.py
https://github.com/nothing14yyx/quant_trade/blob/97952698bdebd012f7baef14230bf69822c2b006/quant_trade/param_search.py