FINANCIAL INDICATOR

VALIDATION ANALYSIS

Technical Analysis for SME Review

Document Purpose:	SME Review and Decision on Indicator Implementation
Prepared For:	Saleem Ahmad (Subject Matter Expert)
Prepared By:	AI Trading System Development Team
Date:	December 10, 2025
Version:	1.0
Classification:	Internal - Technical Review

1. EXECUTIVE SUMMARY

This document presents the validation analysis of 24 technical indicators implemented in our trading system. Saleem's validation testing revealed discrepancies between our calculated values and industry-standard benchmarks (pandas_ta library). This document explains the root causes and presents options for resolution.

Metric	Value	Status
Total Indicators Tested	24	
Indicators Passed	11	PASS
Indicators Failed	13	REQUIRES REVIEW
Success Rate	45.8%	
Primary Root Cause	Smoothing Method Differences	

2. DETAILED VALIDATION RESULTS

The following table shows each indicator's validation status, the maximum difference observed, and the identified root cause:

Indicator	Status	Max Diff	Root Cause
RSI (14)	FAIL	42%	SMA vs Wilder's RMA smoothing
ADX (14)	FAIL	Large	Standard EMA vs Wilder's RMA
+DI / -DI	FAIL	Large	Standard EMA vs Wilder's RMA
ATR (14)	FAIL	Varies	SMA vs Wilder's RMA
Stochastic %K	FAIL	Varies	Fast (raw) vs Slow (smoothed)
Stochastic %D	FAIL	Varies	Upstream %K difference
ROC	FAIL	Varies	Period 12 vs Test Period 10
Ichimoku Senkou A	FAIL	Small	Missing 26-period forward shift
Ichimoku Senkou B	FAIL	Small	Missing 26-period forward shift
Bollinger Upper	FAIL	Small	Sample vs Population std
Bollinger Lower	FAIL	Small	Sample vs Population std
BB Width	FAIL	Small	Upstream band differences
CCI	FAIL	Small	Mean deviation calculation
SMA (20/50/200)	PASS	<0.1%	Matches standard
EMA (12/26/50)	PASS	<0.1%	Matches standard
MACD	PASS	<0.1%	Matches standard
MACD Signal	PASS	<0.1%	Matches standard
Bollinger Middle	PASS	<0.1%	Uses SMA correctly
Williams %R	PASS	<0.1%	Matches standard
Momentum	PASS	<0.1%	Matches standard
OBV	PASS	<0.1%	Matches standard
KAMA	PASS	<0.1%	Matches standard
TRIX	PASS	<0.1%	Matches standard

3. ROOT CAUSE ANALYSIS

3.1 Wilder's Smoothing vs Simple Moving Average

The primary cause of indicator discrepancies is the difference between Wilder's Smoothing (also called Relative Moving Average or RMA) and Simple Moving Average (SMA). J. Welles Wilder Jr., creator of RSI, ATR, and ADX, specified a particular smoothing method that gives more weight to recent values.

Our Current Implementation (SMA-based):

```
gain = delta.where(delta > 0, 0).rolling(window=period).mean()
loss = (-delta).where(delta < 0, 0).rolling(window=period).mean()
```

Industry Standard (Wilder's RMA):

```
gain = delta.where(delta > 0, 0).ewm(alpha=1/period, adjust=False).mean()
loss = (-delta).where(delta < 0, 0).ewm(alpha=1/period, adjust=False).mean()
```

Key Differences:

Aspect	SMA (Our Current)	Wilder's RMA (Industry Standard)
Formula	sum(values) / n	alpha * current + (1-alpha) * previous
Alpha	N/A	1/period (e.g., 1/14 for RSI)
Weight Distribution	Equal weight to all values	Exponentially decaying weights
Responsiveness	Slower to react	Faster to react to recent changes
Used By	Some academic models	TradingView, Bloomberg, pandas_ta

3.2 Stochastic Oscillator Variants

There are multiple variants of the Stochastic Oscillator. Our implementation uses the 'Fast' variant, while most platforms default to the 'Slow' variant.

Variant	%K Calculation	%D Calculation	Used By
Fast (Our Current)	Raw formula	SMA of raw %K	Some day traders
Slow (Industry Std)	SMA(3) of raw %K	SMA(3) of smoothed %K	TradingView, Most platforms
Full	SMA(n) of raw %K	SMA(m) of smoothed %K	Configurable platforms

3.3 Ichimoku Cloud Projection

The Ichimoku Cloud's Senkou Span A and Senkou Span B lines are designed to be projected FORWARD in time by 26 periods. This creates the 'cloud' that shows future support/resistance levels. Our current implementation calculates these values at the current candle without the forward shift.

Our Current Implementation:

```
senkou_a = (tenkan + kijun) / 2 # Calculated at current candle
senkou_b = (high_52 + low_52) / 2 # Calculated at current candle
```

Industry Standard:

```
senkou_a = ((tenkan + kijun) / 2).shift(26) # Shifted 26 periods forward
senkou_b = ((high_52 + low_52) / 2).shift(26) # Shifted 26 periods forward
```

Note: The shift creates a 'cloud' that projects into the future, helping traders anticipate support/resistance. Without the shift, the cloud loses its predictive value.

3.4 Other Minor Differences

Indicator	Our Implementation	Industry Standard	Impact
Bollinger Bands	ddof=1 (sample std)	ddof=0 (population std)	Very small (<0.5%)
ROC	Period = 12	Often Period = 10	Test config mismatch
CCI	Standard mean deviation	Some use median	Small difference

4. OPTIONS FOR SME DECISION

The following options are presented for Saleem's review and decision. Each option has different trade-offs in terms of industry compatibility, development effort, and flexibility.

OPTION A: Match Industry Standards (RECOMMENDED)

Update all indicator calculations to match industry-standard implementations used by TradingView, Bloomberg, and pandas_ta library.

Indicator	Change Required	Complexity
RSI	Change from SMA to Wilder's RMA (ewm alpha=1/14)	Low
ATR	Change from SMA to Wilder's RMA (ewm alpha=1/14)	Low
ADX	Change smoothing to Wilder's RMA for all components	Medium
+DI / -DI	Change smoothing to Wilder's RMA	Medium
Stochastic	Add %K smoothing (SMA of 3) for Slow variant	Low
Ichimoku	Add .shift(26) to Senkou A and B	Low
Bollinger	Change ddof=1 to ddof=0 (optional)	Very Low

Pros:

- Values will match TradingView charts exactly

- Industry-standard implementation increases credibility

- Users can cross-reference with other platforms

- pandas_ta validation will pass at >95%

Cons:

- Requires re-backfilling all historical data

- 2-3 days development and testing effort

OPTION B: Document as Custom Implementation

Keep current implementation but document it as a 'custom' or 'modified' calculation method. This approach acknowledges the differences while explaining the rationale.

Pros:

- No code changes required

- No data re-backfill needed

- SMA-based RSI is valid (just different)

Cons:

- Values won't match TradingView or other platforms
- May confuse users expecting standard values
- Documentation overhead for explaining differences
- Cannot use standard trading strategies without adjustment

OPTION C: Add Configurable Parameters

Implement both calculation methods and allow users to select which variant they prefer. Store both values or calculate on-demand based on user preference.

Pros:

- Maximum flexibility for users
- Can support multiple trading strategies
- Research-friendly (compare methods)

Cons:

- Most complex implementation
- Nearly doubles storage requirements if storing both
- UI complexity for user selection
- 1-2 weeks development effort

5. RECOMMENDATION

Based on our analysis, we strongly recommend **OPTION A: Match Industry Standards** for the following reasons:

1. **User Expectations:** Traders expect RSI 30 to mean oversold based on Wilder's original formula. Our SMA-based RSI produces values up to 42% different, which could lead to incorrect trading decisions.

2. **Cross-Platform Compatibility:** Users will inevitably compare our charts to TradingView. Matching values builds trust and allows users to validate our platform.

3. **AI/ML Training:** If training AI models on our data, non-standard indicators may produce models that don't generalize to real-world trading scenarios.

4. **Industry Credibility:** A professional trading platform should implement standard calculations. Custom implementations require significant documentation and may raise questions.

5. **Moderate Effort:** The changes are straightforward - primarily replacing .rolling().mean() with .ewm(alpha=1/period).mean() in key functions. Estimated 1-2 days coding + 1 day testing.

6. IMPLEMENTATION PLAN (IF OPTION A APPROVED)

Step	Task	Estimated Time
1	Create indicator_calculations_v2.py with corrected formulas	4 hours
2	Unit test each indicator against pandas_ta library	4 hours
3	Update backfill scripts to use new calculation module	2 hours
4	Clear existing data and re-run backfill for all 106 stocks	8-12 hours
5	Re-run backfill for 50 crypto assets	6-8 hours
6	Validation testing with Saleem's validation script	2 hours
7	Deploy updated Cloud Functions	2 hours
	TOTAL ESTIMATED TIME	**28-34 hours**

7. SME DECISION AND SIGN-OFF

Please review the options above and indicate your decision:

- ☐ Option A: Match Industry Standards (Recommended)
- ☐ Option B: Document as Custom Implementation
- ☐ Option C: Add Configurable Parameters
- ☐ Other (please specify below)

Additional Comments / Special Instructions:

SME Name: _____ Date: _____

Signature: _____

APPENDIX A: WILDER'S SMOOTHING FORMULA

For reference, here is the mathematical definition of Wilder's Smoothing (RMA):

Initial Value (first period):

```
RMA_1 = SMA(values, period)
```

Subsequent Values:

```
RMA_n = (previous_RMA * (period - 1) + current_value) / period

Or equivalently using exponential weighting:
RMA_n = alpha * current_value + (1 - alpha) * previous_RMA
where alpha = 1/period
```

This is equivalent to pandas: ***ewm(alpha=1/period, adjust=False).mean()***

APPENDIX B: AFFECTED INDICATORS AND FORMULAS

Indicator	Current Formula	Corrected Formula
RSI	`gain.rolling(14).mean()`	`gain.ewm(alpha=1/14, adjust=False).mean()`
ATR	`tr.rolling(14).mean()`	`tr.ewm(alpha=1/14, adjust=False).mean()`
ADX Smoothing	`EMA calculation`	`ewm(alpha=1/14, adjust=False)`
Stochastic %K	`raw_k (no smoothing)`	`raw_k.rolling(3).mean()`
Senkou Span A	`value (no shift)`	`value.shift(26)`
Senkou Span B	`value (no shift)`	`value.shift(26)`