

# End-to-End AI-Enhanced Regime-Aware Hybrid Alpha Strategy Pipeline [17-Step Workflow]

By Dr. Ray Islam  
Adjunct Professor, IST  
George Mason University, Fairfax, VA, USA

## Overall Objective

An end-to-end hybrid trading strategy prototype that integrates market-based **alpha signals** (trend/momentum + mean-reversion) with FinBERT news sentiment, uses regime-aware gating to combine signals, applies realistic constraints and execution costs, and produces backtest results and reporting artifacts. This pipeline is designed to evolve from a single-ticker MVP (AAPL) into a scalable multi-asset system.

## Overall Goals

- Create reproducible data ingestion and feature pipelines (raw > cleaned > features > feature store).
- Generate three alpha sleeves (trend, reversion, sentiment) and combine them using regime-aware gating.
- Translate alpha into tradable weights with constraints, position sizing, and execution-cost assumptions.
- Backtest weekly rebalancing long-only strategy and produce metrics + tear sheet + logs.
- Lay the groundwork for ML ranker/gating and walk-forward evaluation as future iterations.

## Scripts:

### Run Order

1. script\_01\_define\_and\_ingest.py (Steps 1–5)
2. script\_02\_market\_features.py (Step 6)
3. script\_03\_sentiment\_features.py (Steps 7–9)
4. script\_04\_alphas\_regime\_gating.py (Steps 10–12)
5. script\_05\_portfolio\_execution.py (Steps 13–15)
6. script\_06\_backtest\_report\_prod.py (Steps 16–17)

## 17 Steps Checklist (with Script Mapping)

Step	What you do	Script	Primary Output Artifact
1	Define scope, universe, objective, and assumptions	Script 1	data/processed/strategy_spec.json
2	Ingest OHLCV daily bars (Polygon)	Script 1	data/raw/<TICKER>_ohlcw_daily_raw.csv
3	Ingest company news feed (Polygon reference news)	Script 1	data/raw/<TICKER>_news_raw.csv
4	Market data validation & cleaning (OHLCV sanity checks)	Script 1	data/processed/<TICKER>_ohlcw_daily_clean.csv
5	News validation & cleaning (dedup, normalize dates, drop invalid)	Script 1	data/processed/<TICKER>_news_clean.csv
6	Feature engineering: returns panel (returns, vol, liquidity, ranges)	Script 2	data/features/<TICKER>_returns_panel.parquet
7	FinBERT sentiment scoring on news text	Script 3	data/sentiment/<TICKER>_news_scored.csv
8	Aggregate sentiment to daily features (mean/sum/count)	Script 3	data/sentiment/<TICKER>_sentiment_daily.csv
9	Build feature store (merge market features +	Script 3	data/features/<TICKER>_feature_store.parquet

	sentiment features)		
10	Construct alpha sleeves: trend, mean-reversion, sentiment (standardized)	Script 4	data/features/<TICKER>_alphas_regime_gated.parquet
11	Regime detection (e.g., volatility-based risk_on / risk_off)	Script 4	regime labels in the same parquet
12	Gating/ensemble weighting: combine sleeves based on regime	Script 4	alpha_combined in the same parquet
13	Constraints & risk controls (liquidity, exposure caps, turnover caps)	Script 5	liquidity_ok + constrained targets
14	Position sizing (sigmoid + volatility targeting; clip to bounds)	Script 5	target_weight_raw
15	Execution model (cost bps incl. vol bump; used during backtest)	Script 5	exec_cost_bps
16	Backtesting: weekly rebalance, turnover cap, apply costs; walk-forward scaffold	Script 6	data/backtests/<TICKER>_backtest.csv
17	Reporting + production hygiene: tear sheet PDF, metrics JSON, logs, kill switch	Script 6	data/reports/<TICKER>_tear_sheet.pdf + logs

## Notes / Tips

- If running inside Jupyter/IPython, do NOT use `__file__`. Use `Path.cwd()` or a `resolve_base_dir()` helper.
- If you see an 'xformers not installed' message, it is a warning (performance), not a failure.
- If pandas raises an aggregation `TypeError`, use the tuple-aggregation syntax:  
`sent_mean=('sentiment_score','mean')`.
- Set environment variable `KILL_SWITCH=1` to safely stop execution in Scripts 5–6 (production hygiene).

## Expected Outputs

### **strategy\_spec.json**

Purpose: Stores the strategy definition (ticker, date range, objective).

How to read: Open in any text editor. Verify ticker/start/end match your intended run.

### **\*\_ohlcv\_daily\_raw.csv / \*\_ohlcv\_daily\_clean.csv**

Purpose: Raw and cleaned daily market data used for feature engineering/backtest.

How to read: Open in Excel/Pandas. Confirm date continuity, `close>0`, `volume>=0`, and no duplicates.

### **\*\_news\_raw.csv / \*\_news\_clean.csv**

Purpose: Raw and cleaned news articles pulled from Polygon.

How to read: Check `published_utc/date` fields. Ensure titles exist. Verify duplicates removed.

### **\*\_returns\_panel.parquet**

Purpose: Market feature panel containing returns/volatility/liquidity measures.

How to read: Load with `pandas.read_parquet()`. Inspect columns: `ret_1d`, `vol_21d`, `adv_dollars_20d`, `fwd_ret_5d`.

**\*\_news\_scored.csv**

Purpose: News articles with FinBERT sentiment labels/scores.

How to read: Review sentiment\_label and sentiment\_score. Spot-check a few titles vs scores for reasonableness.

**\*\_sentiment\_daily.csv**

Purpose: Daily aggregated sentiment features (mean/sum/count).

How to read: Look for non-zero sent\_count days; compare sent\_mean vs major news dates.

**\*\_feature\_store.parquet**

Purpose: Master dataset for modeling: market features + sentiment features aligned by date.

How to read: Ensure no look-ahead (sentiment uses same-day news). Confirm no NaNs in key fields.

**\*\_alphas\_regime\_gated.parquet**

Purpose: Alpha sleeves + regime labels + combined alpha after gating.

How to read: Plot alpha\_trend/alpha\_reversion/alpha\_sentiment and regime. Confirm alpha\_combined changes with regime.

**\*\_targets\_with\_costs.parquet**

Purpose: Portfolio targets with constraints + execution-cost parameters.

How to read: Inspect target\_weight\_raw (0..1), liquidity\_ok flag, and exec\_cost\_bps distribution.

**\*\_backtest.csv**

Purpose: Day-by-day backtest results including NAV, position weights, turnover, costs.

How to read: Plot nav over time; check turnover spikes align with rebalance dates; verify trade\_cost is applied.

**\*\_metrics.json**

Purpose: Summary performance metrics.

How to read: Open JSON; compare annual\_return, annual\_vol, sharpe, max\_drawdown; validate costs/turnover.

**\*\_tear\_sheet.pdf**

Purpose: Visual tear sheet report (equity curve, drawdown, rolling sharpe, weights).

How to read: Confirm equity curve consistency, drawdown realism, and rolling sharpe stability. Use this as your review artifact.

**logs/strategy\_run.log**

Purpose: Execution log for debugging and auditing.

How to read: Search for WARN/ERROR. Confirm script stages ran fully and outputs were saved.