📈TikTok Real-Time Trend & Prediction System – Multi-Agent / LLM Continuous Improvement Roadmap

Goal: Build a *living* TikTok trend intelligence engine in Cursor that (1) ingests real-time data from **EnsembleData**, (2) runs an **Analyzer Agent** pipeline (feature + metrics extraction), (3) applies **LLM Insight Agents** for semantic enrichment & hypothesis generation, (4) trains & refines **prediction models**, and (5) continuously self-evaluates & improves via a **Feedback & Orchestration Agent**.

> **Format:** Pure step-by-step procedural roadmap (no code) optimized for Cursor task execution. You can later turn each numbered step into a task / file / agent config.

🖨️High-Level Architecture (Mental Model)

1. **Data Layer:** EnsembleData → Raw Ingestion Buffer → Normalized Store (Postgres / DuckDB / Parquet) → Feature Store.
2. **Analyzer Agent Cluster:** Deterministic workers that compute quantitative metrics (velocity, momentum, decay, engagement efficiency, novelty, creator dispersion, retention curve approximations).
3. **Semantic & Insight LLM Agents:** Enrich with topic clustering, emerging theme detection, narrative labels, anomaly rationales, synthetic test cases.
4. **Predictor Models:** Classical (LogReg / XGBoost) + Time-series (Temporal Fusion / simple LSTM) + Optional lightweight RL ranking.
5. **Feedback Loop:** Evaluation Agent compares predictions vs realized outcomes; triggers retraining or feature revision tasks; maintains a changelog.
6. **Governance & Observability:** Metrics dashboards, data quality checks, drift detection, prompt performance tracking.
7. **Continuous Improvement Cycle:** (Collect → Analyze → Predict → Evaluate → Adjust) automated on a daily cadence, with intra-day micro-updates.

🔗Phase 0 – Foundation & Definitions

Objective: Establish exact scopes & success metrics. 1. Define *Viral Success Label*: e.g., hashtag reaches Top 5% growth in views within 24h after first detection window. 2. Define *Prediction Horizon*: (e.g., predict virality 6h → 24h ahead). 3. Define *Update Cadences*: Ingestion (15 min), Metrics (hourly), Predictions (hourly), Evaluation (daily), Model retrain (weekly or drift-triggered). 4. Draft Data Dictionary (raw → normalized → features). 5. Create a "Change Log" file for every structural / feature / model update.

🔗Phase 1 – Data Ingestion & Normalization

Objective: Reliable, schema-stable raw + curated data. 1. List initial entity types: `Hashtag`, `Video`, `Creator`, `Music/Sound`. 2. For each entity, list mandatory fields (id, timestamp, counts) & optional fields. 3. Set ingestion frequency (cron or workflow) using EnsembleData endpoints (no code yet; note endpoints to invoke later). 4. Design staging schema: Raw JSON blobs + extracted atomic columns. 5. Define *Idempotency Rules* (dedupe by video_id + fetch_timestamp window). 6. Specify retention policy (raw 30 days, aggregated indefinite or 1 year). 7. Plan *Latency Budget* (target <5 min from fetch to feature availability). 8. Draft Data Quality Checks (DQC): missing fields %, monotonic view count, outlier spikes threshold.

🔗Phase 2 – Core Quantitative Metrics (Analyzer Agent)

Objective: Deterministic metrics the LLM can later interpret. 1. For each hashtag each run: compute *Volume*, *New Videos Count*, *Unique Creators*, *Total Views Increment*, *Engagement Rates* (likes/views, comments/views, shares/views). 2. Compute *Velocity*: (current window volume − previous window) / previous window. 3. Compute *Acceleration*: (current velocity − previous velocity) / |previous velocity|. 4. Compute *Momentum Score*: weighted combination (e.g., 0.4 velocity + 0.3 acceleration + 0.2 engagement efficiency + 0.1 creator dispersion). 5. Compute *Novelty Index*: inverse overlap of top creators/sounds vs last N hours baseline. 6. Compute *Persistence / Half-Life*: hours for engagement rate to drop 50% (approx via exponential fit over successive snapshots). 7. Store each metric snapshot in a time-series table. 8. Tag top *Emerging* hashtags (Momentum above dynamic percentile & Novelty above threshold).

🔗Phase 3 – Semantic / LLM Enrichment

Objective: Convert raw metrics & tokens into higher-level concepts. 1. Cluster hashtags + caption keywords (use vector store later; for now define label fields: `cluster_id`, `theme_label`). 2. Use LLM Insight Agent to propose *Theme Labels* (constraints: 2–3 words, stable vocabulary file). 3. Generate *Trend Narratives*: short bullet reasons why momentum is high (inputs: metrics + top creators + overlapping sounds). 4. Detect *Cross-Theme Collisions*: LLM compares new emerging hashtags vs existing clusters; flag if semantic distance < threshold yet metrics diverge. 5. Maintain *LLM Prompt Registry*: version each prompt template; store output acceptance rate. 6. Add *Risk / Noise Flags*: LLM inspects anomalies (sudden view spikes with low creator dispersion) to mark probable artificial boosting. 7. Produce *Synthetic Counterfactuals*: LLM generates hypothetical variations (e.g., "What if creator dispersion doubles?") for feature stress tests.

🔗Phase 4 – Feature Engineering & Store

Objective: Stable feature sets for models. 1. Enumerate Feature Groups: (A) Temporal Growth (velocity, acceleration, lagged values), (B) Engagement Ratios, (C) Creator Diversity, (D) Semantic Category (encoded), (E) Novelty Indices, (F) Past Prediction Error residuals. 2. Define *Feature Freshness SLA* (e.g., all features computed within 10 min of ingestion for current window). 3. Create *Feature Versioning Plan*: feature_set_v1, v2… with change notes. 4. Implement *Missing Data Strategy* (forward-fill limited, else sentinel / median impute spec). 5. Mark *Leaky Features* (future information) to explicitly exclude. 6. Draft *Feature Importance Tracking Plan* (store importance per retrain cycle).

🔗Phase 5 – Baseline Prediction Models

Objective: Initial predictive capability. 1. Define training window (e.g., rolling 14 days) + validation strategy (time-based split). 2. Label creation specification (viral= top 5% cumulative views growth within 24h after detection timestamp). 3. Choose baseline algorithms: Logistic Regression (calibrated probabilities) + XGBoost (nonlinear interactions). 4. Define evaluation metrics: AUC, PR-AUC (imbalanced), Early Precision (precision@k for top predicted hashtags), Calibration (Brier score), Lead Time (avg hours before actual surge). 5. Draft *Model Storage Spec*: model id, training dataset hash, feature list, metrics snapshot. 6. Set *Promotion Criteria*: deploy if PR-AUC improves ≥5% & calibration shift <0.02. 7. Document *Fallback Behavior*: if new model underperforms live by threshold over N hours → auto rollback.

🔗Phase 6 – Advanced Modeling & Ranking

Objective: Improve lead time & stability. 1. Add Time-Series Component: per-hashtag sequence of momentum features → simple RNN/LSTM spec (define sequence length, sampling cadence). 2. Evaluate Ensemble: blend (0.6 XGBoost prob + 0.4 RNN prob) or stacking spec. 3. Introduce *Uncertainty Estimation*: quantile regression / Monte Carlo dropout spec. 4. Define *Explainer Outputs*: SHAP summary fields stored per prediction cycle. 5. Create *Prediction Ranking Rules* (filter low-n noise, penalize low novelty, boost cross-cluster spread potential).

🔗Phase 7 – Continuous Evaluation Loop (Feedback Agent)

Objective: Automatic learning & quality assurance. 1. Daily alignment job: compare predicted top K vs actual top K realized. 2. Compute *Attribution of Misses*: categorize misses (late detection, mis-labeled cluster, feature gap, model drift). 3. Generate *LLM Error Analysis Summary*: feed metrics + misses; produce prioritized improvement tasks. 4. Maintain *Drift Monitors*: feature distribution KL divergence, label prevalence shift; trigger retrain tasks. 5. Maintain *Prompt Performance Metrics*: acceptance rate, hallucination incidents (flag by rule-based checks). 6. Weekly retrospective summary file appended automatically.

🔗Phase 8 – Orchestration & Agents Role Definition

Objective: Clear responsibilities for each agent. 1. **Ingestion Agent**: orchestrates API pulls, validates schema, logs volumes. 2. **Metric Agent**: computes new quantitative metrics; updates Feature Store. 3. **Semantic Agent**: runs clustering + narrative labeling. 4. **Prediction Agent**: loads latest model; scores; writes predictions. 5. **Evaluation Agent**: compares realized outcomes; updates KPIs. 6. **Improvement Agent**: digests evaluation + LLM analysis; creates backlog tasks. 7. **Governance Agent**: checks policy constraints (rate limit usage, PII avoidance, license compliance). 8. **Notification Agent**: sends alerts (emerging trend, model drift, pipeline failure). 9. **Scheduler**: defines cadence table (document this matrix explicitly).

🔗Phase 9 – Observability & Logging

Objective: Detect failures early & maintain trust. 1. Define *Core Health Metrics*: ingestion latency, feature freshness, prediction coverage, drift scores, alert count. 2. Set thresholds & escalation steps (warning vs critical). 3. Log taxonomy: `EVENT_TYPE` (INGEST_START, METRIC_DONE, MODEL_PROMOTED, DRIFT_ALERT, PROMPT_UPDATE). 4. Daily automated status summary (human-readable bullet list) for quick scan.

🔗Phase 10 – Human-in-the-Loop & Guardrails

Objective: Ensure LLM outputs remain grounded. 1. Define *Verification Rules* (e.g., LLM cannot assert numerical metrics not present in analytic data; must cite metric ids or timestamps). 2. Add *Sanity Check Heuristics* (e.g., engagement rate > 1.0 → auto flag & discard narrative). 3. Maintain *Blocked Phrases / Low-Value Labels* list to prune generic outputs ("awesome trend", "very viral"). 4. Collect manual feedback labels (ACCEPT / EDIT / REJECT) on narratives to improve prompt iteration schedule.

🔗Phase 11 – Continuous Improvement Cadence

Objective: Institutionalize iteration. 1. Daily: ingestion, metrics, predictions, evaluation summary. 2. Twice Weekly: prompt refinement cycle (Semantic Agent prompts version bump). 3. Weekly: model drift check & potential retrain. 4. Biweekly: feature set review; prune or add features based on SHAP & error analysis. 5. Monthly: architecture audit (latency, cost, accuracy vs goals).

🔗Phase 12 – Extension Paths

Optional Enhancements: 1. Cross-Platform Correlation (add YouTube Shorts, IG Reels) for early multi-platform signals. 2. Product Trend Layer (integrate EchoTik for shop SKU performance → enrich prediction features). 3. Reinforcement Learning Re-Ranker (optimize ranking for user engagement on your internal dashboard clicks). 4. Active Learning Loop (uncertain predictions prioritized for manual label review to refine dataset). 5. A/B Testing Framework (serve different narrative prompt versions; measure user dwell time / action rate).

🕐Success KPI Dashboard (Define Early)

- Lead Time Gain (hours before spike identified).
- Precision@K (top 10 predicted vs realized trending hashtags in next 12–24h).
- Recall@K (coverage of actual future viral items in predictions set).
- Mean Calibration Error.
- False Surge Rate (predicted viral but didn't exceed threshold).
- LLM Narrative Acceptance Rate.
- Data Latency (median minutes ingestion → prediction availability).

🧰Backlog Template (for each improvement task)

Title: Short action. **Category:** Feature / Model / Prompt / Infra / Data Quality. **Trigger Source:** Drift / Miss Analysis / Manual Feedback / LLM Suggestion. **Hypothesis:** Why change helps. **Success Metric:** Target delta. **Owner Agent:** (Improvement / Metric / Semantic ...) **Review Date:** Next evaluation cycle.

📝Initial Action Order (Turn This Into Tasks Today)

1. Finalize viral label & horizon definitions.
2. Draft staging & feature schemas (document).
3. Specify ingestion cadence + endpoint list + rate budget sheet.
4. Enumerate first metric formulas (write spec).
5. Define cluster labeling prompt template v1.
6. Define evaluation metric formulas & thresholds.
7. Create agent role matrix & scheduling table.
8. Draft governance + verification rules file.
9. Set initial KPI dashboard metric list & layout sketch.
10. Draft improvement backlog with at least 5 seed tasks.

Ready for the next step? I can now: (a) generate the agent role matrix table, (b) create a scheduling cadence matrix, or (c) supply initial prompt templates. Just tell me which you want next.