Multi-Model Regime-Aware Trading Framework

Overview

This framework defines a hierarchical, multi-model system for forecasting, regime detection, and adaptive trade decision-making. The objective is to predict short-term (15-minute horizon) market movements, quantify model coherence, and dynamically determine when to trust the collective signal to execute trades.

Model Architecture

Model (1): Cumulative Return Forecaster

Task: Point forecast the 15-minute ahead cumulative return — both direction and magnitude.
Output: Predicted signed cumulative return (■_cumret_15m).
Purpose: Primary directional signal for expected market move.

Model (2): Volatility Regime Classifier

Task: Predict the volatility regime for the next 15 minutes.
Classes: Low, Medium, High volatility.
Purpose: Identifies the current risk environment, informing whether trend signals are reliable or the market is unstable.

Model (3): Drift Regime Classifier

Task: Predict average drift (directional bias) over the next 15 minutes.
Classes: Positive High Drift, Negative High Drift, Low Drift.
Purpose: Measures directional persistence and supports alignment with Model (1)’s forecast.

Model (4): Directional Accuracy Forecaster

Task: Predict 15-minute ahead directional accuracy using normalized price data and all engineered features.
Output: Probability that a directional forecast (up or down) will be correct.
Purpose: Confirms or rejects Model (1)’s direction using a broader feature context.

Trade Decision Logic

A trade is executed only when:

1. Model (1) and Model (4) agree on the direction of the next 15-minute move.
2. Model (2) and Model (3) indicate compatible regimes for that trade type:
- For a long trade: high positive drift, low volatility.
- For a short trade: high negative drift, low volatility.

This enforces directional consensus and regime validation before entry.

Extended Regime Models

To enrich regime awareness, additional models are trained on alternative market segments:

1. High / Low Volume Segments — detect liquidity and participation regimes (**Model 5**).
2. Order Flow Imbalance Segments — identify persistent buy/sell pressure (**Model 6**).

Each segment produces specialized predictive models whose outputs complement Models (1)–(4) and can be factored into the decision logic stage.

Meta-Model for Signal Reliability

After simulating trades based on the above rule set, the trade outcomes (PnL, directional success, etc.) are used to train a meta-model:

- Model (7): Logistic Regression or Decision Tree
- Task: Learn when to trust the composite signals from Models (1)–(6).
- Target: Binary trade success (1 = profitable / correct, 0 = not).
- Purpose: Adaptively decide whether to act on the ensemble's signal.

Coherence Quantification via PCA

The outputs of all models (1–6) are aggregated into a feature matrix of model predictions. A Principal Component Analysis (PCA) model is applied to this matrix to quantify how well model outputs collectively explain market variance.

- Target: Future returns (e.g., realized 15-minute return).
- Metric: Variance explained by the first principal component (PC1).
- Interpretation:
• High PC1 variance $\rightarrow$ strong model agreement and coherent market structure.
• Low PC1 variance $\rightarrow$ model disagreement, chaotic regime, unreliable signals.

Final Decision Layer

The PCA-derived coherence metric is combined with the meta-model (LR or Decision Tree) to create a final signal filter.

- Input: PCA variance ratio and outputs of Models (1)–(6).
- Output: "Trade / No Trade" decision — indicating when the ensemble's consensus is trustworthy enough to act.

Summary Pipeline

1. Train Models (1)–(4) on base features and 15-min targets.
2. Generate trade signals based on agreement and regime filters.
3. Extend with volume and order-flow imbalance segment models.
4. Train a meta-model on trade outcomes to learn signal trustworthiness.
5. Apply PCA to (1)-(6) model outputs to measure predictive coherence.
6. Combine PCA output with the meta-model to form a final decision filter for trade execution.

Strategy Implementation

1. Optimise trade size, stop loss and take profit logic.
2. Optimise thresholds and parameters with risk constraints.
3. Create a live signal dashboard for manual execution.
4. Develop a live execution pipeline connected to TopStep orders API (Live data from Databento).