Project: Autonomous AI-Driven Insider Trading Signal Discovery

Overview

This project aims to build a fully autonomous, closed-loop financial modeling system that discovers, validates, and evolves predictive trading features, specifically for identifying potential insider trading activity in real time using SEC Form 4 filings and market microstructure data. The architecture integrates GPT-5-level agentic planning, self-supervised learning, symbolic regression, and similarity-based pattern recall using vector databases.

Problem Statement

We want to: - Build a repeatable, verifiable trading strategy that detects anomalous trading behavior potentially indicative of insider activity. - Ingest and parse SEC Form 4 filings (insider purchases/sales) to act as anchoring events. - For each filing, look back historically to extract: - Abnormal lit and dark pool volumes - Short sale volume and price suppression activity - Bid-ask spread anomalies or microstructure signatures - Store extracted features and signals in a vector database so similar patterns can be retrieved. - Train models to recognize such signals in real-time *before* a Form 4 is filed.

Core System Architecture

1. Data Ingestion

- **SEC Filings**: Use `sec-edgar-api` or `edgartools` to ingest Form 4 filings.
- **Market Microstructure**: Include volume, trade prints, dark pool (ATS), TRF, short sale data.
- **LOB Snapshots**: Use DeepLOB-ready formats where available.
- **Index Filing Events**: Store CIK, symbol, transaction date, type (e.g. code P/S/M), price, and shares.

2. Feature Generation Pipeline

Modular Generators: - **Self-Supervised Embeddings**: TS2Vec, DeepLOB, Series2Vec - **Symbolic Regression**: PySR to distill deep features into explicit equations - **Topological Data Analysis**: Persistent homology to encode cycles, voids before events - **Change Point Detection**: BOCPD, CUSUM, HMMs to detect regime shifts - **Anomaly Proxies**: OFI imbalance, quote stuffing, abnormal spread z-scores, short intensity

Window Configuration: - Lookback: 5–60 days before filing - Event anchors: Filing date, transaction date

3. Vector Database (Similarity Retrieval)

- **Indexing**: Store feature vectors and embeddings (e.g. TS2Vec, symbolic, topological)
- **DBs**: FAISS, Milvus, Pinecone
- **Metadata**: timestamp, symbol, feature type, regime label

- **Usage**: On new data, embed signal and find k-NN matches to past insider-like anomalies

4. Backtesting & Labeling

- Define labels as: Form 4 purchase/sale occurs within T days after a time window
- Compute Sharpe, drawdown, PnL curves on signals generated from embeddings
- Segment performance by market regime, asset class, volatility bucket

5. GPT-5 Agentic Planning Layer

- Reads feature logs, embeddings, and backtest diagnostics
- Generates:
- Feature transformations
- Symbolic forms from latent vectors
- Retraining triggers (e.g. drift detected)
- Pipeline edits (e.g. new TDA operator, HMM change)
- Uses tool APIs:
- `run_backtest()`
- `generate_feature()`
- `refresh_vector_db()`
- `retrain_model()`
- Can iterate over failures, propose repairs or test theories

6. Feedback Loop & Continual Learning

- Model is retrained periodically or on regime change
- Feature candidates are filtered using cross-regime validation
- GPT observes decay in live feature performance and responds

Key Decisions

- Use vector embeddings for pattern memory and pre-filing similarity detection
- Use GPT agent to plan, reason, and evolve the system (not just run pipelines)
- Use symbolic distillation to improve transparency and reduce overfitting
- Reward loop using Sharpe, Sortino, and drawdown-based metrics
- Evaluate features across time-based folds and regimes for survivability

Technologies & Libraries

- **SEC Parsing**: `sec-edgar-api`, `edgartools`, lxml
- **Market Data**: yfinance, polygon.io, FINRA short sale / ATS data
- **Feature Learning**: TS2Vec, DeepLOB, PySR, ruptures, tick.hawkes
- **Vector DB**: FAISS / Pinecone with pgvector or metadata overlay
- **Backtest**: vectorbt, bt, zipline (modular)
- **Agent Framework**: LangGraph, OpenAgents, AutoGen (to control GPT)

Use Cases

- Insider anomaly detection and prediction
- Market microstructure regime shift detection
- Automated signal fingerprint clustering
- Hybrid symbolic + vector strategy audits

What Comes Next

- Set up GitHub repo and define folders for ingestion, features, agent, model, vector DB
- Define first prompt chain for GPT to analyze backtest logs and propose retraining
- Build vector database with embeddings from past Form 4 anchored windows
- Validate vector similarity queries (e.g. “find days similar to when insider bought at suppressed bid”)
- Schedule retrain loop and feature pruning every N days

This file serves as a blueprint. It can be shared across ChatGPT conversations to maintain system continuity and inform any agent of your full system vision, decisions, and objectives.