End-to-End Forecasting Pipeline — Clean Workflow

Step 1 — Data sources & schema contract

Input: Market choice from Step 0.

Output:

- docs/01_sources.md — public datasets/endpoints, update frequency, caveats
- config/schema.yaml — strict column names, units, timezone rules

Where it lives: Repo

Used by: Steps 2–4 (ingestion, QA, features)

LLM usage: ChatGPT 5.2 — List reliable public datasets and define a clean schema.

Step 2 — Ingestion (idempotent)

Input: config/market.yaml, config/schema.yaml, data endpoints

Output:

- src/ingest/*.py — pullers and normalization
- data/raw/{run_id}/*.parquet (or CSV)

Where it lives: Code in repo, raw data gitignored

Used by: Step 3 QA

LLM usage: Claude Opus 4.5 — Implement ingestion scripts matching the schema.

Step 3 — QA gate (hard pass/fail)

Input: Raw data + schema rules

Output:

- src/qa/* — QA checks
- reports/qa/{run_id}_qa.json / .md
- data/clean/{run_id}/*.parquet

Used by: Step 4 (only if QA passes)

LLM usage: Claude Opus 4.5 — Add missingness, duplicates, DST, range checks.

Step 4 — Feature engineering

Input: Clean data + feature spec

Output:

- src/features/*

- data/features/{run_id}.parquet
- Optional feature spec markdown

Used by: Modeling and validation

LLM usage: Claude Opus 4.5 — Deterministic feature pipeline aligned to targets.

Step 5 — Baseline model (benchmark)

Input: Feature dataset

Output:

- src/models/baseline.py
- outputs/preds_baseline_{run_id}.csv
- reports/metrics/baseline_{run_id}.json

LLM usage: Claude Opus 4.5 — Transparent baseline with rolling backtests.

Step 6 — Improved model (best)

Input: Features + baseline results

Output:

- src/models/model.py (LightGBM/XGBoost)
- models/model_{run_id}.bin
- Predictions and metrics

LLM usage: Claude Opus 4.5 — Time-series CV, early stopping, feature importance.

Step 7 — Validation & stress tests

Input: Baseline + improved model outputs

Output:

- src/validation/*
- reports/figures/*
- reports/validation/{run_id}.md

LLM workflow: Codex diagnoses failures, Opus applies fixes.

Step 8 — Prompt-curve translation (forecast → trade)

Input: Forecast outputs

Output:

- src/trading/*
- outputs/signal_{run_id}.csv

- reports/trading/{run_id}.md

LLM usage: Gemini 3 Pro — Define positioning framework and invalidation rules.

Step 9 — Programmatic LLM commentary

Input: QA stats, metrics deltas, forecast changes

Output:

- src/reporting/llm_commentary.py
- reports/commentary/{run_id}.md
- reports/llm_logs/{run_id}.json

LLM usage: ChatGPT designs prompt, Opus implements script.

Step 10 — Engineering hardening

Output:

- README.md, requirements.txt, Makefile / CLI
- python -m pipeline run --date ...
- tests/*

LLM workflow: Opus implements, Codex audits reproducibility.

Step 11 — Final report assembly

Input: QA summaries, metrics, plots, trading example, LLM commentary

Output:

- report/report.md
- report/report.pdf

LLM usage: ChatGPT 5.2 — Produce a tight 1–3 page FAANG-style report.