(Programming) Building a Real-Time Crypto AI Service

- Due Dec 5 by 11:59pm
- Points 100
- Submitting a text entry box, a website url, a media recording, or a file upload

Programming Assignment: Building a Real-Time Crypto AI Service

You'll work in teams of five to transform one of your individual volatility detection models into a real-time AI service. Across four weeks, your team will design, build, deploy, and monitor a system that can:

- Stream live data from Coinbase (via Kafka)
- Process and predict in real time using FastAPI
- Track models with MLflow
- Monitor performance using Prometheus, Grafana, and Evidently

Assignment Overview

Your deliverables will combine engineering depth with operational excellence—just as you'd expect in an enterprise AI environment.

Weekly Plan

Week 4 – System Setup & API Thin Slice

Goal: Build your first working system prototype in replay mode (not live yet).

Tasks:
• Choose your base or composite model.
• Draw a simple system diagram (ingestor → features → API → monitoring).
• Create /health, /predict, /version, and /metrics endpoints in FastAPI.
• Launch Kafka (KRaft) and MLflow using Docker Compose.

• Replay a 10-minute dataset to test your pipeline.
• Write two docs: team_charter.md (roles) and selection_rationale.md (model choice).

Deliverables:
• docker-compose.yaml, Dockerfiles, and architecture diagram
• Working /predict endpoint (with sample curl)
• Team charter + selection rationale

Week 5 – CI, Testing & Resilience

Goal: Add testing and reliability to your system.

Tasks:
• Set up CI with GitHub Actions (Black/Ruff + one integration test).
• Add reconnect, retry, and graceful shutdown to Kafka services.
• Write a load test (100 burst requests).
• Use .env.example for config and secrets.

Deliverables:
• Passing CI pipeline (screenshot or badge)
• Load test + brief latency report
• Updated README (≤10-line setup guide)

Week 6 – Monitoring, SLOs & Drift

Goal: Add dashboards, alerts, and drift detection.

Tasks:
• Integrate Prometheus metrics (latency, request count, errors, consumer lag)
• Create Grafana dashboards (p50/p95 latency, error rate, freshness)
• Define SLOs (p95 ≤ 800 ms – aspirational target)
• Schedule Evidently drift report in docs/drift_summary.md
• Add rollback toggle via MODEL_VARIANT=ml|baseline

Deliverables:
• Grafana dashboard (JSON + screenshot)
• Evidently report + summary
• SLOs (docs/slo.md) and Runbook (docs/runbook.md)

Week 7 – Demo, Handoff & Reflection

Goal: Demonstrate and hand off your full working system.

Tasks:
• Record 8-min demo showing startup, prediction, failure recovery, rollback.
• Write concise runbook (startup, troubleshooting, recovery).
• Summarize latency, uptime, PR-AUC vs baseline.
• Tag final release.

Deliverables:
• Demo video link (YouTube)
• Final repo with docs and Compose setup

API Contract

POST /predict
Request:

```
{
  "rows": [{"ret_mean": 0.05, "ret_std": 0.01, "n": 50}]
}
```

Response:

```
{
  "scores": [0.74],
  "model_variant": "ml",
  "version": "v1.2",
```

```
  "ts": "2025-11-02T14:33:00Z"
}
```

Supporting Endpoints:
GET /health -> { "status": "ok" }
GET /version -> { "model": "rf_v1", "sha": "abc123" }
GET /metrics -> Prometheus-format metrics

Grading Rubric

Area	Points	What's Evaluated
Architecture & Setup	25	One-command startup, clear diagram, working endpoints
CI/CD & Reliability	25	Passing CI pipeline, fault tolerance, basic load testing
Monitoring & Drift	30	Functional Prometheus + Grafana, Evidently drift detection, rollback feature
Demo & Professionalism	20	Clear demo, readable documentation, clean repo

Submission Summary

Submit via GitHub (tagged release):
• Source code, docker-compose.yaml, and docs

• README (≤10-line setup)
• Demo video link (unlisted YouTube/Loom)

Run command:
`docker compose up -d`
`curl -X POST http://localhost:8000/predict ...`

Clarifications & Tips

- Redpanda may be used as a full Kafka replacement.
- CI minimum = lint + one replay test.
- p95 ≤ 800 ms is aspirational, not mandatory.
- Example failure: restart Kafka broker or simulate API 500 error.
- Demo checklist provided for transparency.