bitcoin-timeseries-ml-engineering

Public

p... 1 Branch 0 Tags Go to file t Go to file Add file Code

FabioVinelli Harden: time-series dataloader rigor + recruiter docs 8c6733d · 5 minutes ago 2 Commits

private	Portfolio hardening: public scaffold	34 minutes ago
src	Harden: time-series dataloader rigor ...	5 minutes ago
.env.example	Portfolio hardening: public scaffold	34 minutes ago
.gitignore	Harden: time-series dataloader rigor ...	5 minutes ago
AGENT.md	Harden: time-series dataloader rigor ...	5 minutes ago
MAIN_DEV_NOTES.md	Portfolio hardening: public scaffold	34 minutes ago
README.md	Harden: time-series dataloader rigor ...	5 minutes ago

README

bitcoin-timeseries-ml-engineering

Bitcoin Time-Series ML Engineering (LSTM + Walk-Forward Validation) — portfolio-grade pipeline with private alpha redaction

> **Portfolio project for the IBM AI Engineering Certificate**
> This repo demonstrates end-to-end ML engineering for financial time-series (data processing → model training → evaluation → reproducible runs).
> **Important boundary:** my proprietary trading datasets, QFL-DCA rules, and "alpha discoveries" are intentionally **not** published.

What this repo is

A **production-style ML pipeline** for Bitcoin forecasting experiments using:

- **Time-series-safe splits** (chronological + walk-forward evaluation)
- **Leakage-resistant scaling** (fit on train only)
- **Model architectures** (LSTM variants + optional attention/CNN)
- **Evaluation** (standard ML metrics + trading-aware metrics)

This is designed to be **auditable, reproducible, and recruiter-readable**.

What this repo is NOT

- Not a "copy-paste profitable strategy"
- Not a data dump of my historical trades
- Not a release of my full QFL-DCA alpha rulebook
- Not financial advice

If you want to reproduce results, you must use **your own datasets** (or the optional public sample described below).

Privacy / Alpha Redaction Policy (Explicit)

To protect years of research and private trading data, the following are **excluded from this public repo**:

- Private `data/` (raw, processed, or labeled datasets)
- Private `outputs/` (full experiment result tables, matched trades, correlations)
- Certain strategy-specific configurations / thresholds

About

No description, website, or topics provided.

Readme
Activity
0 stars
0 watching
0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages

Python 100.0%

Suggested workflows

Based on your tech stack

SLSA Generic generator — Configure
Generate SLSA3 provenance for your existing release workflows

Django — Configure
Build and Test a Django Project

Python package — Configure
Create and test a Python package on multiple Python versions.

More workflows Dismiss suggestions

- Full QFL-DCA optimization documents or proprietary rule sets

What is included instead:

- The **pipeline** and **engineering rigor**
- Config-driven training/evaluation
- A clear "Bring Your Own Data" interface
- Optional **synthetic or public sample** dataset support (recommended)

Repo Structure (public-safe)

```
.
├── src/
│   ├── data/            # feature engineering + processor (train-only scaling)
│   ├── models/          # LSTM architectures
│   ├── training/        # trainer + metrics + walk-forward
│   └── utils/           # inference utilities
├── tests/               # unit tests (pipeline sanity checks)
├── notebooks/           # exploration / colab (sanitized)
├── configs/             # PUBLIC configs (no private thresholds)
├── docs/                # methodology notes (portfolio narrative)
├── scripts/             # helper scripts (sanitized)
├── main.py              # CLI entry point (train/evaluate)
├── requirements.txt
├── README.md
└── AGENT.md             # contributor/agent runbook (hardening + release flow)
```

Engineering Rigor Highlights (what recruiters should notice)

- **Reproducibility:** seed control + config-driven runs (`config.yaml`)
  (see seed handling in `main.py`)
- **Leakage control:** scaling is fit on train split only (zero-leakage design)
- **Time-series validation:** walk-forward cross-validation (no random k-fold)
- **Time-series rigor:** DataLoaders use `shuffle=False` to preserve temporal order
- **Checkpointing:** best model is saved and reloaded for evaluation
- **Metrics:** ML metrics + trading-aware metrics (Sharpe, drawdown, directional accuracy)

Quick Start (Bring Your Own Data)

1) Install

```
python -m venv .venv
source .venv/bin/activate
pip install -r requirements.txt
```

2) Provide your BTC data

You provide a CSV at minimum containing OHLCV daily bars.

Expected location (default):

- `data/raw/btc_ohlcv_daily.csv` (not committed)

Optional:

- `data/raw/btc_onchain.csv` (not committed)

> If you don't want to use private data, use the optional "public sample mode" once you add it (see below).

3) Train / Evaluate

```
python main.py --mode train --config config.yaml
python main.py --mode evaluate --config config.yaml
```

Outputs (local only, not committed):

- `models/best_model.pt`
- `models/scalers.joblib`
- `outputs/metrics.json` / `outputs/metrics.txt`
- `outputs/test_predictions.csv`

Public Sample Mode (Recommended)

To keep this repo reproducible without exposing private data:

- Add a **small public dataset** (or generate synthetic data) under:
  - `data/sample/` (committed)
- Keep real datasets in:
  - `data/raw/` (gitignored)

This keeps the project runnable for reviewers while protecting your edge.

Results (How to interpret)

This repo focuses on **engineering quality** and correct evaluation for noisy markets.

Important:

- Any "paper numbers" (e.g., R^2=0.991) are **literature-reported benchmarks**, not guaranteed reproduction.
- Your actual results depend heavily on data window, feature set, regime shifts, and cost assumptions.

Safety / Compliance

This is educational and for portfolio demonstration only. No financial advice. Use at your own risk.

License

Choose a permissive license for portfolio visibility (MIT/Apache-2.0). If you want to restrict use, choose a more protective license. (Decide before first push.)

Contact

If you're a recruiter/hiring manager: this repo is intended to show ML engineering skill in time-series forecasting + evaluation discipline.