CryptoCopilot

راهنمای کامل پروژه

یک دستیار معاملاتی ارزِ دیجیتالِ چندزبانه و کاملاً آزمایشی — ترکیبِ یادگیری ماشین، تحلیل تکنیکال، فاندامنتال و چتِ مستند RAG در یک نظرِ واحد و قابل‌توضیح.

راهنمای فنیِ کامل — یادگیری ماشین، بک‌اند، فرانت‌اند و هوش مصنوعی

ابزارِ کمک‌به‌تصمیم، نه توصیه‌ی مالی · فقط معامله‌ی آزمایشی

نسخه‌ی ۱.۰ · ژوئن ۲۰۲۶

فهرست مطالب

فهرست مطالب 2
1. CryptoCopilot چیست 4
دامنه‌ی صادقانه — مهندسی در اولویت 4
2. نگاهِ کلی: یک معماریِ چندزبانه 5
چرا این شکل (تصمیم‌های طراحی) 6
مالکیتِ جدول‌ها — دقیقاً یک نویسنده برای هر جدول 6
3. لایه‌ی داده — کوین‌ها، منابع و پایگاه‌داده 6
ده کوین، و چرا این‌ها 6
منابعِ داده — همگی عمومی و رایگان، به‌همراه نشانی 7
مدلِ پایگاه‌داده 7
4. سرویسِ یادگیری ماشین (پایتون) 9
گام ۱ — برداشتِ داده (Ingestion) 9
گام ۲ — مهندسیِ ویژگی (تبدیلِ کندل به سیگنال) 9
گام ۳ — هدف و تقسیم‌بندی (تعریفِ مسئله) 10
گام ۴ — XGBoost (مدل)، توضیح داده‌شده 10
گام ۵ — کالیبراسیونِ ایزوتونیک (صادق‌کردنِ احتمال‌ها) 11
گام ۶ — SHAP (توضیحِ هر پیش‌بینی) 11
گام ۷ — ارزیابی: واقعاً چقدر خوب است؟ 11
ROC-AUC — متریکِ سرتیتر، با توضیحِ کامل 11
سایرِ متریک‌ها 11
5. سرویسِ بک‌اند (جاوا / Spring Boot) 13
Spring AI — چیست و چگونه استفاده می‌کنیم 13
ta4j — اندیکاتورهای تحلیل تکنیکال در جاوا 13
کتابخانه‌های جاوای استفاده‌شده، و چرا 14
6. لایه‌ی هوش مصنوعی — Embeddingها و RAG (پژوهشگر) 16
Embedding چیست؟ (از پایه) 16
RAG چیست، و چرا از آن استفاده کنیم؟ 16
چرا pgvector؟ 16
Ollama در برابرِ OpenAI — چرا هر دو وجود دارند (کلیدِ تعویض) 16
Embeddingها: nomic-embed-text در برابرِ text-embedding-3-small 17

7. تجمیع‌کننده‌ی تحلیل‌گر (Analyst) 17
سلامتِ فاندامنتالِ دو-لایه 17
نگهبانِ توهم (Hallucination Guard) 18
8. موتورِ معامله‌ی آزمایشی (Paper Trading) 18
یک «پُرشدن (fill)» چگونه کار می‌کند 19
متریک‌های عملکرد 19
بک‌تست — یک نتیجه‌ی صادقانه 19
9. فرانت‌اند (React) 21
انتخاب‌های فناوری 21
نمودارها 21
صفحه‌ها 21
10. خطِ لوله‌ی درخواستی: ML API (پایتون / FastAPI) 23
11. واژه‌نامه — مفاهیمِ کلیدی به‌زبانِ ساده 23
کریپتو و بازار 23
یادگیری ماشین 23
هوش مصنوعی / RAG 23
معاملات 24
12. اجرا و نتایجِ صادقانه 24
چگونه کلِ سامانه را اجرا کنیم 24
نتایجِ صادقانه، در یک جدول 24

1. CryptoCopilot چیست

CryptoCopilot یک دستیار شخصی برای کسی است که تازه وارد دنیای ارز دیجیتال شده و می‌خواهد در درکِ بازار کمک بگیرد. این برنامه به یک پرسشِ متمرکز برای یک معامله‌گرِ خُرد پاسخ می‌دهد:

> «با توجه به هر اتفاقی که هم‌اکنون در بازار و اخبار در حال رخ‌دادن است، درباره‌ی این کوین‌ها چه باید فکر کنم — و اگر معامله‌ای انجام دهم، عملکردش چگونه خواهد بود؟»

این برنامه برای پاسخ‌دادن، هر یک از ۱۰ کوین را هم‌زمان از چهار زاویه‌ی مختلف بررسی می‌کند و سپس آن‌ها را در یک نظرِ واحدِ روشن و قابل‌توضیح ترکیب می‌کند:

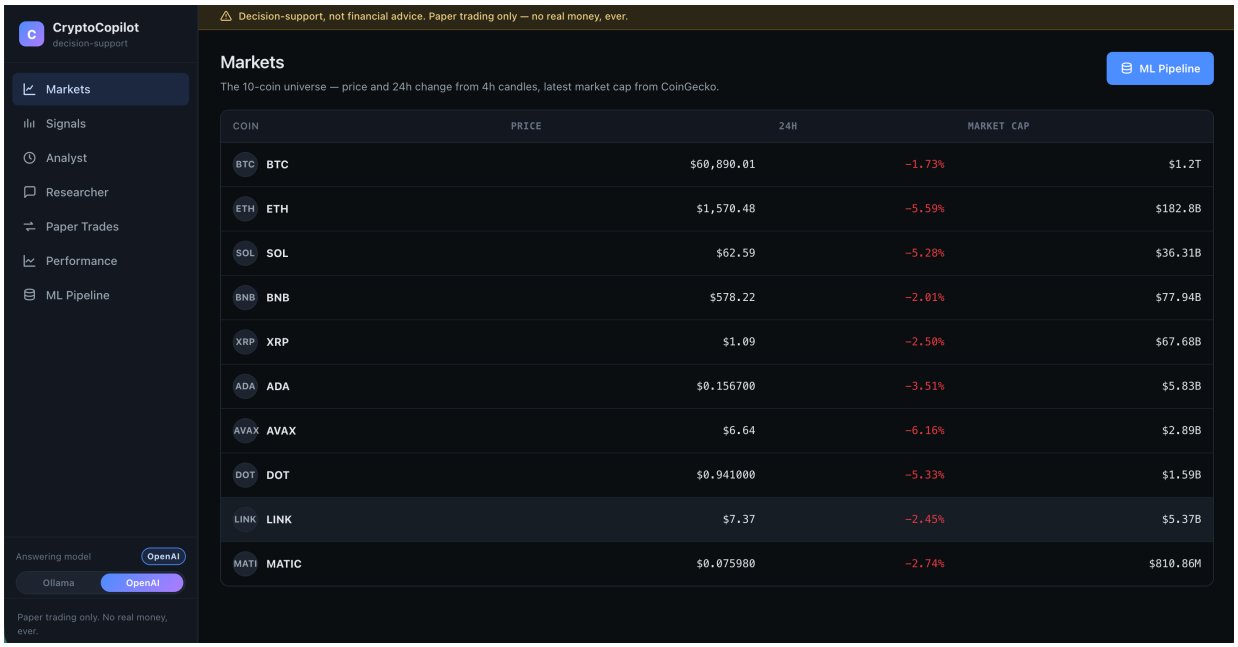
- یادگیری ماشین (ML) — مدلی که احتمالِ بالا رفتن، پایین آمدن یا ثابت ماندنِ قیمت در ۲۴ ساعتِ آینده را تخمین می‌زند.
- تحلیل تکنیکال (TA) — خوانشی قانون‌محور از اندیکاتورهای کلاسیکِ نمودار (ایچیموکو، MACD، RSI، بولینگر، ATR).
- فاندامنتال — بررسیِ سلامتِ کوین بر پایه‌ی فعالیتِ آن‌چین و داده‌های جامعه و توسعه‌دهندگان.
- اخبار + چتِ RAG — یک «پژوهشگر» که فقط با استفاده از منابعِ واقعی و ارجاع‌دار (اخبار، داده‌ی آن‌چین و یک پایگاه‌دانشِ تدوین‌شده) به پرسش‌ها پاسخ می‌دهد.

بخشی به نام تحلیل‌گر (Analyst) این چهار دیدگاه را در یک حکمِ واحد ترکیب می‌کند — یک جهت، یک سطحِ قطعیت و یک امتیازِ توافق. سپس کاربر می‌تواند با ثبتِ یک معامله‌ی آزمایشی (با پولِ فرضی) عمل کند. هر عدد ردیابی‌پذیر است، هر معامله ثبت می‌شود و یک سلبِ مسئولیت در همه‌ی صفحه‌ها نمایش داده می‌شود.

دامنه‌ی صادقانه — مهندسی در اولویت

این پروژه عمداً یک پروژه‌ی نمونه‌کار و یادگیری است. هدف شکست‌دادنِ بازار نیست (هیچ مدلِ کوچکی نمی‌تواند به‌طور قابل‌اتکا چنین کند). هدف، نمایشِ مهندسیِ تمیز و در سطحِ تولید در چند زبان و چند حوزه است: خطوطِ لوله‌ی داده، یادگیری ماشین، یک بک‌اندِ جاوا، هوش مصنوعیِ مبتنی بر بازیابی، و یک فرانت‌اندِ مدرنِ وب. نتایج صادقانه گزارش می‌شوند — برای مثال، امتیازِ مدلِ ML عمداً متواضعانه است و دقیقاً توضیح می‌دهیم چرا.

> ⚠️ ابزارِ کمک‌به‌تصمیم، نه توصیه‌ی مالی. فقط معامله‌ی آزمایشی — هرگز پولِ واقعی. فقط ارز دیجیتال؛ بدونِ فروشِ استقراضی، بدونِ اهرم.



تصویر ۱ — صفحه‌ی Markets: مجموعه‌ی ۱۰ کوین به‌همراه قیمتِ زنده و تغییرِ ۲۴ ساعته. نوارِ کناری (چپ) همه‌ی صفحه‌ها را فهرست می‌کند؛ کلیدِ «Answering model» (پایین-چپ) هوش مصنوعی را میانِ Ollamaی محلی و OpenAI جابه‌جا می‌کند.

2. نگاهِ کلی: یک معماریِ چندزبانه

«چندزبانه» یعنی سامانه از بیش از یک زبانِ برنامه‌نویسی ساخته شده و هر زبان جایی به‌کار رفته که در آن قوی‌ترین است. CryptoCopilot به‌صورتِ پنج کانتینرِ داکر پیرامونِ یک پایگاه‌داده‌ی مشترک اجرا می‌شود:

کانتینر	زبان / فناوری	وظیفه
db	Postgres 16 + pgvector	تنها منبعِ حقیقت — نگه‌دارنده‌ی همه‌ی داده‌ها و ایندکسِ برداریِ هوش مصنوعی.
ml	Python (کارگرِ دسته‌ای)	طبقِ زمان‌بندی بیدار می‌شود، داده می‌گیرد، مدلِ ML را اجرا می‌کند، نتیجه را می‌نویسد و می‌خوابد.
ml-api	Python (FastAPI)	امکانِ اجرای دستیِ ingest / train / predict را می‌دهد (همان کدِ ml).
backend	Java + Spring Boot	مغزِ برنامه: REST API، تحلیل تکنیکال، چتِ RAG، تحلیل‌گر و معامله‌ی آزمایشی.
frontend	React + nginx	وب‌سایتی که در مرورگر می‌بینید — یک کلاینتِ نازک روی بک‌اند.

نمودارِ متنیِ زیر نشان می‌دهد این‌ها چگونه به هم وصل می‌شوند. مهم‌ترین ایده، فلشِ ورودی به پایگاه‌داده از دو زبان به‌طورِ هم‌زمان است:

```
browser  →  frontend (React + nginx)
                 │  REST / JSON  (same origin; nginx proxies /api)
                 ▼
            backend (Spring Boot)  ── MODULAR MONOLITH (ta4j · Spring AI · trading · analyst)
                 │ JDBC            │ HTTP  /api/ml/* → ingest·train·predict
                 ▼                 ▼
 (writes preds)  db (Postgres + pgvector)     ml-api (Python · FastAPI)
      ▲                                          │  (same code, shared model dir)
 ml (Python batch worker) ──────────────────────┘  writes predictions / raw data
```

چرا این شکل (تصمیم‌های طراحی)

پایگاه‌داده، مرزِ میانِ دو زبان است. پایتون جدول‌های خود را می‌نویسد؛ جاوا فقط آن‌ها را می‌خوانَد. هیچ فراخوانیِ مستقیمی از جاوا به مدلِ پایتون وجود ندارد، هیچ فایلِ مدلِ مشترکی نیست و هیچ «فراخوانیِ راه‌دور» شبکه‌ای میانِ آن‌ها رخ نمی‌دهد. این دو صرفاً از طریقِ جدول‌های SQL همکاری می‌کنند. این رویکرد ساده، پایدار و آسان‌فهم است — حتی اگر سرویسِ پایتون خاموش باشد، برنامه همچنان آخرین نتایجی را که نوشته سرو می‌کند.

بک‌اند یک «مونولیتِ ماژولار» است، نه میکروسرویس. این یک برنامه‌ی جاوا با ماژول‌های داخلیِ تمیز است (تحلیل تکنیکال، RAG، معاملات، تحلیل‌گر). شکستنِ آن به ده‌ها سرویسِ کوچک، دردِسرهای عملیاتیِ فراوانی (کشفِ سرویس، فراخوانیِ شبکه، تراکنش‌های توزیع‌شده) برای حلِ مشکلاتِ مقیاسی اضافه می‌کرد که یک پروژه‌ی تک‌کاربره اساساً ندارد.

ML یک کارِ دسته‌ای است، نه یک سرویسِ زنده‌ی وب. جهتِ قیمت در افقِ ۲۴ ساعته به‌کندی تغییر می‌کند، پس کانتینرِ `ml` طبقِ زمان‌بندی بیدار می‌شود، کارش را انجام می‌دهد، در پایگاه‌داده می‌نویسد و می‌خوابد. کانتینرِ خواهرِ آن، `ml-api`، یک لایه‌ی نازکِ وب روی همان کد قرار می‌دهد تا همان کارها بتوانند به‌صورتِ درخواستی از صفحه‌ی «ML Pipeline» در رابطِ کاربری نیز اجرا شوند.

مالکیتِ جدول‌ها — دقیقاً یک نویسنده برای هر جدول

برای تمیز ماندنِ پایگاه‌داده‌ی مشترک، هر جدول دقیقاً یک نویسنده دارد. جاوا هرگز جدول‌های پایتون را نمی‌نویسد؛ پایتون هرگز جدول‌های جاوا را نمی‌نویسد.

مالک	این جدول‌ها را می‌نویسد	می‌خوانَد
ml (پایتون)	ohlcv, market_meta, news, onchain, fundamentals, predictions, prediction_drivers	جدول‌های خودش
backend (جاوا)	account_state, positions, trades, orders, و vector_store (Spring AI)	همه‌ی جدول‌های ml (فقط‌خواندنی)
frontend (ری‌اکت)	—	فقط REST بک‌اند

3. لایه‌ی داده — کوین‌ها، منابع و پایگاه‌داده

ده کوین، و چرا این‌ها

این مجموعه شاملِ ۱۰ ارزِ دیجیتالِ بزرگ و نقدشونده است — آن‌هایی که عمیق‌ترین تاریخچه‌ی بازار، بیشترین پوششِ خبری و بیشترین داده‌ی آن‌چین را دارند. همین عمق دقیقاً چیزی است که یک پروژه‌ی داده و ML نیاز دارد. این‌ها طراحی‌های متفاوتی را نیز پوشش می‌دهند، که پروژه را جذاب‌تر از ۱۰ کوینِ تقریباً یکسان می‌کند:

نماد	نام	چیست / چرا در مجموعه است
BTC	بیت‌کوین	نخستین ارز دیجیتال؛ اثباتِ کار (PoW)؛ معیارِ بازار و تنها کوینی که اینجا سری زمانیِ روزانه‌ی آن‌چینِ غنی دارد.
ETH	اتریوم	بزرگ‌ترین پلتفرمِ قراردادِ هوشمند؛ اثباتِ سهام (PoS)؛ بستر بیشترِ DeFi و NFT.
SOL	سولانا	یک زنجیره‌ی PoS با توانِ عبورِ بالا؛ سریع و ارزان؛ رقیبِ مهمِ اتریوم.
BNB	بی‌ان‌بی	توکنِ کاربردیِ اکوسیستمِ BNB؛ بزرگ و نقدشونده.
XRP	ریپل	متمرکز بر پرداخت‌های فرامرزیِ سریع و ارزان؛ طراحیِ اجماعِ متمایز.
ADA	کاردانو	یک پلتفرمِ PoS با رویکردِ پژوهش‌محور.

AVAX	آوالانچ	یک پلتفرمِ سریعِ PoS با طراحیِ چندزنجیره‌ای («ساب‌نت»).
DOT	پولکادات	شبکه‌ی هم‌کنش‌پذیری که زنجیره‌های بسیاری را به هم وصل می‌کند («پاراچین»).
LINK	چین‌لینک	شبکه‌ی پیشروِ اوراکل — داده‌ی دنیای واقعی را به قراردادهای هوشمند می‌رسانَد.
MATIC/POL	پالیگان	شبکه‌ی مقیاس‌پذیریِ اتریوم. (MATIC در اواخرِ ۲۰۲۴ به POL تغییرِ نام داد؛ لودر هر دو را زیرِ MATIC می‌دوزد.)

منابعِ داده — همگی عمومی و رایگان، به‌همراه نشانی

هر منبع، عمومی و رایگان است. یک قانونِ اصلی («بدونِ نقطه‌ی شکستِ واحد») یعنی اگر هر منبعی از کار بیفتد یا پولی شود، خطِ لوله مشکل را ثبت می‌کند و از آن می‌گذرد — هرگز کرش نمی‌کند.

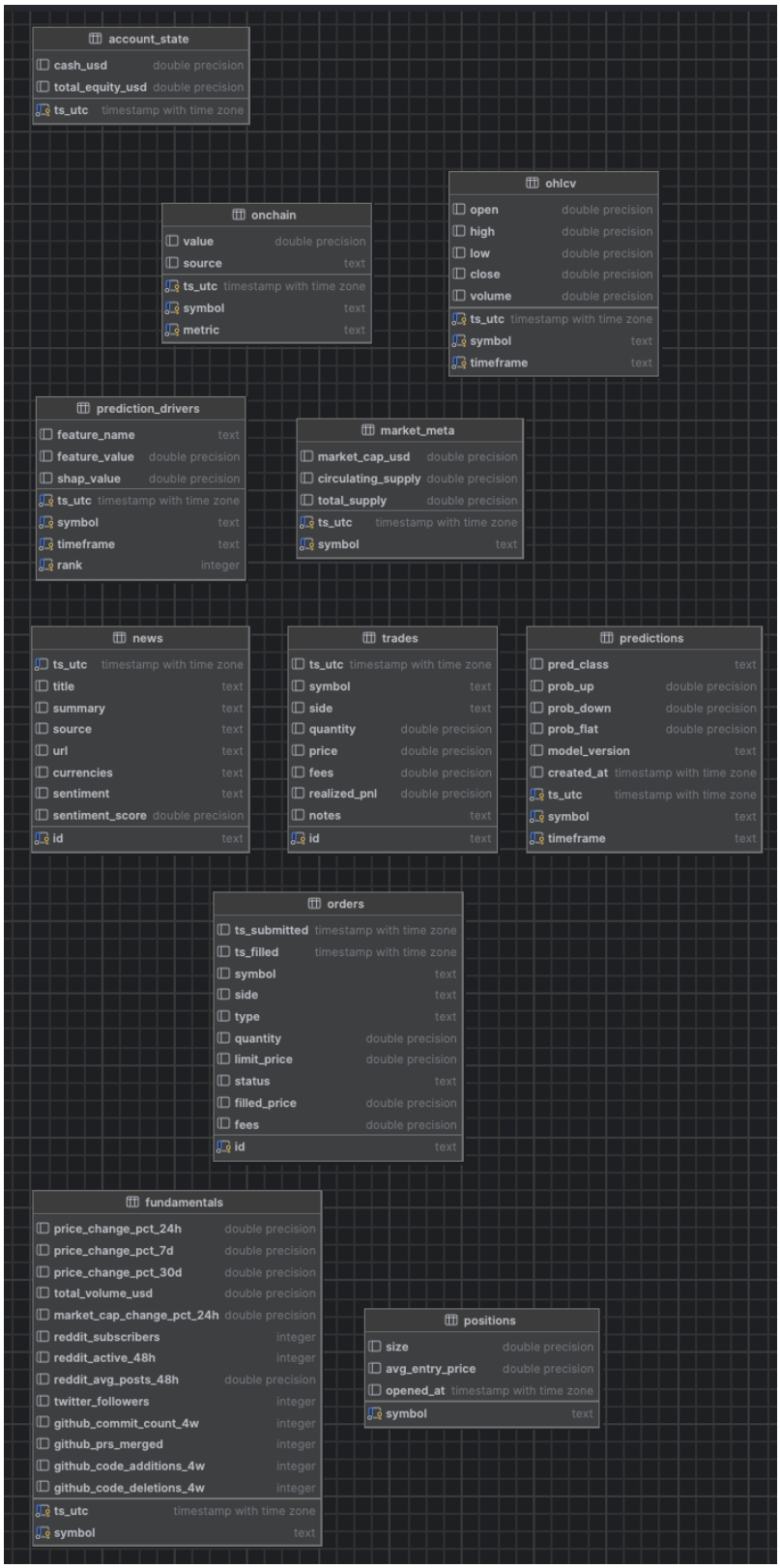
منبع	برای	نشانی (URL)	احراز
Binance public API	کندل‌های OHLCV (۱h / ۴h / ۱d، حدود ۲ سال)	binance.com/api	ندارد
CoinGecko Demo	ارزش بازار، عرضه، داده‌ی جامعه/توسعه/بازار (هر ۱۰)	coingecko.com/en/api	کلیدِ رایگان (۱۰k/ماه، ۳۰/دقیقه)
CoinDesk RSS	تیترهای خبری (پنجره‌ی ۱۸۰ روزه)	coindesk.com/arc/outboundfeeds/rss/	ندارد
Cointelegraph RSS	تیترهای خبری	cointelegraph.com/rss	ندارد
Decrypt RSS	تیترهای خبری	decrypt.co/feed	ندارد
The Block RSS	تیترهای خبری	theblock.co/rss.xml	ندارد
Bitcoin Magazine RSS	تیترهای خبری	/bitcoinmagazine.com/.rss/full	ندارد
Blockchain.com Charts	متریک‌های آن‌چینِ BTC	blockchain.com/charts	ندارد
Etherscan	متریک‌های آن‌چینِ ETH	etherscan.io/apis	کلیدِ رایگان (۵/ثانیه، ۱۰۰k/روز)
پایگاه‌دانشِ تدوین‌شده	سازوکار/توکنومیکسِ کوین (نوشته‌شده در ریپو)	—	—

معنای «OHLCV»: برای هر بازه‌ی زمانی (یک «کندل»)، قیمتِ بازگشایی (O)، بیشترین (H)، کمترین (L) و بسته‌شدن (C) به‌علاوه‌ی حجمِ (V) معامله‌شده را ذخیره می‌کنیم. این، ماده‌ی خامِ هم ویژگی‌های ML و هم اندیکاتورهای تحلیل تکنیکال است.

نخستین برداشتِ کاملِ داده ۲۳۱,۰۸۲ ردیف بارگذاری کرد: `ohlcv` حدود ۲۲۶,۲۰۰ (۱۰ کوین × ۳ تایم‌فریم × حدود ۲ سال)، `market_meta` ۳,۶۶۰، `news` ۱۲۴ (پنجره‌ی متحرک)، `onchain` ۱,۰۸۸ (BTC + ETH) و `fundamentals` ۱۰ (یک تصویرِ لحظه‌ایِ تازه برای هر کوین).

مدلِ پایگاه‌داده

همه‌ی داده‌ها در Postgres (یک پایگاه‌داده‌ی رابطه‌ای) قرار دارند. دو گروهِ تقریباً جدا از جدول‌ها — متعلق به پایتون (داده + ML) و متعلق به جاوا (حسابِ معامله‌ی آزمایشی) — یک اسکیمای مشترک را که در `db/init.sql` تعریف شده به اشتراک می‌گذارند. نمودارِ زیر همه‌ی جدول‌ها و ستون‌های آن‌ها را نشان می‌دهد.



تصویر ۲ — اسکیمای پایگاه‌داده. جدول‌های متعلق به پایتون (بالا): ohlcv، market_meta، news، onchain، fundamentals، predictions، prediction_drivers. متعلق به جاوا (پایین): account_state، positions، trades، orders. Spring AI جدولِ vector_store را می‌افزاید.

جدول	مالک	هر ردیف چه چیزی نگه می‌دارد

ohlcv	ml	یک کندلِ قیمت: زمان، نماد، تایم‌فریم، قیمت‌های open/high/low/close و حجم.
market_meta	ml	ارزش بازار و عرضه‌ی در گردش / کل در یک لحظه‌ی زمانی.
news	ml	یک خبر: تیتر، خلاصه، منبع، نشانی، کوین‌های برچسب‌خورده و برچسب + امتیازِ احساسات.
onchain	ml	مقدارِ یک متریکِ آن‌چین (مثلاً آدرس‌های فعال) برای یک نماد در یک زمان، با منبعش.
fundamentals	ml	تصویرِ جامعه + توسعه + بازار: تغییرِ ۳۰d/۷d/۲۴h، حجم، فعالیتِ ردیت/توییتر/گیت‌هاب.
predictions	ml	آخرین پیش‌بینیِ مدل برای هر کوین: کلاسِ پیش‌بینی + احتمالِ بالا/پایین/ثابت.
prediction_drivers	ml	سه دلیلِ برترِ SHAP پشتِ هر پیش‌بینی (نامِ ویژگی، مقدار، سهم).
account_state	backend	تصویرِ لحظه‌ایِ حسابِ آزمایشی: نقدینگی و ارزشِ کل در طولِ زمان (منحنیِ سرمایه).
positions	backend	کوین‌های نگه‌داری‌شده‌ی فعلی: اندازه و میانگینِ قیمتِ ورود.
trades	backend	هر خرید/فروشِ اجراشده با مقدار، قیمت، کارمزد و سود/زیانِ محقق‌شده.
orders	backend	سفارش‌ها: market/limit، وضعیت (pending/filled/cancelled)، قیمتِ پُرشدن.
vector_store	Spring AI	ایندکس جست‌وجوی هوش مصنوعی: تکه‌های متن + بردارهای ۷۶۸) embedding عدد هرکدام).

4. سرویسِ یادگیری ماشین (پایتون)

این قلبِ پروژه است. داستان، در یک خط: تبدیلِ حدود ۲ سال کندلِ خامِ قیمت به یک احتمالِ کالیبره‌شده و قابل‌توضیح که هر کوین در ۲۴ ساعتِ آینده بالا می‌رود، پایین می‌آید یا ثابت می‌مانَد. بیایید هر مرحله را گام‌به‌گام و از پایه مرور کنیم.

گام ۱ — برداشتِ داده (Ingestion)

پایتون در کار با داده عالی است، پس همه‌ی دریافتِ داده اینجاست. کتابخانه‌های استفاده‌شده:

- ccxt — کتابخانه‌ای یکپارچه برای صرافی‌های ارز دیجیتال؛ برای دانلودِ کندل‌های OHLCV باینانس.
- requests — فراخوانیِ ساده‌ی HTTP به Blockchain.com، CoinGecko و Etherscan.
- feedparser — خواندنِ فیدهای خبریِ RSS.
- vaderSentiment — ابزاری سبک و قانون‌محور که هر تیترِ خبری را به‌صورتِ مثبت/منفی/خنثی به‌رایگان و به‌صورتِ محلی برچسب می‌زند (در واژه‌نامه توضیح داده شده).
- pandas / numpy — ابزارهای استانداردِ کار با جدولِ اعداد و ریاضیات.
- SQLAlchemy + psycopg2 — نوشتنِ نتایج در Postgres. APScheduler — زمان‌سنجی که برداشتِ داده را روزانه اجرا می‌کند.

گام ۲ — مهندسیِ ویژگی (تبدیلِ کندل به سیگنال)

یک مدلِ یادگیری ماشین نمی‌تواند مستقیماً از قیمت‌های خام یاد بگیرد؛ باید ویژگی‌ها را محاسبه کنیم — اعدادِ پُرمعنا که وضعیتِ کنونیِ بازار را توصیف می‌کنند. CryptoCopilot برای هر کوین در هر گامِ زمانی ۴۶ ویژگی محاسبه می‌کند، همگی فقط روبه‌گذشته (هرگز به آینده نگاه نمی‌کنند)، از جمله:

- بازده در بازه‌های ۱h، ۴h، ۲۴h و ۷d (قیمت اخیراً چقدر تغییر کرده).
- RSI در دوره‌های ۷/۱۴/۲۱، MACD و تقاطعِ آن، ADX، Stochastic (قدرتِ روند).
- بولینگر B% و پهنای باند، ATR% (نوسان)، نوسانِ محقق‌شده (۷d/۲۴h).
- Z-score حجم (آیا حجم به‌طورِ غیرعادی بالاست؟) و نسبت‌های میانگینِ متحرک.

- ایچیموکو، نوشته‌شده از پایه (Tenkan، Kijun، Senkou A/B، پرچم‌ها و فاصله‌های ابر)، با یک جابه‌جاییِ زمانیِ ایمن در برابرِ نشت.
- ویژگی‌های تقویمی (مثلِ روزِ هفته) و یک کدِ one-hot برای هویتِ کوین.

مهندسیِ ویژگی درونیِ پایتون می‌مانَد (به‌صورتِ فایل‌های Parquet کش می‌شود)؛ فقط پیش‌بینی‌های نهایی از مرزِ پایگاه‌داده به جاوا عبور می‌کنند. تکرارِ کوچک — اینکه جاوا اندیکاتورهای خود را با ta4j بازمحاسبه می‌کند — عمدی است و دو زبان را تمیز جدا نگه می‌دارد.

گام ۳ — هدف و تقسیم‌بندی (تعریفِ مسئله)

مسئله را به یک دسته‌بندیِ ۳-کلاسه تبدیل می‌کنیم. هدف (`y_24h_3class`) هر لحظه را بر اساسِ کاری که قیمت در ۲۴ ساعتِ *بعدی* انجام داده برچسب می‌زند:

- UP اگر بیش از +۲٪ بالا رود، DOWN اگر بیش از −۲٪ پایین آید، در غیرِ این صورت FLAT.

چرا ±۲٪؟ یک باندِ کوچک‌تر (مثلاً ±۱٪) عمدتاً نویزِ تصادفی را می‌گیرد؛ ±۲٪ حرکتی را هدف می‌گیرد که آن‌قدر بزرگ باشد که مهم شود. سپس داده به‌شدت بر اساسِ زمان تقسیم می‌شود — آموزش روی گذشته، اعتبارسنجی روی بُرشی بعدتر، و آزمون روی تازه‌ترین بُرش — با یک فاصله‌ی «قرنطینه»ی ۲۴ ساعته میانِ بُرش‌ها تا هیچ اطلاعاتی از مرز عبور نکند. رعایتِ ترتیبِ زمانی ضروری است: درهم‌ریختنِ ردیف‌ها به مدل اجازه می‌دهد «آینده را ببیند»، که امتیاز را عالی اما بی‌معنا می‌کند.

بُرش	ردیف	پنجره	توازنِ DOWN / FLAT / UP
train	20,961	30-05-2025 → 15-06-2024	0.265 / 0.466 / 0.268
val	5,460	30-08-2025 → 01-06-2025	0.282 / 0.473 / 0.246
test	16,290	30-05-2026 → 01-09-2025	0.202 / 0.537 / 0.261

گام ۴ — XGBoost (مدل)، توضیح داده‌شده

مدلِ اصلی XGBoost است. برای درکِ آن، سه ایده را روی هم بسازید:

- درختِ تصمیم یک فلوچارتِ پرسش‌های بله/خیر درباره‌ی ویژگی‌هاست (مثلاً «آیا RSI زیرِ ۳۰ است؟») که به یک پیش‌بینی ختم می‌شود. یک درخت به‌تنهایی ضعیف است و مستعدِ بیش‌برازش.
- بوستینگ (Boosting) درخت‌های کوچکِ بسیاری را پشتِ سرِ هم آموزش می‌دهد، که هر درختِ تازه روی اصلاحِ خطاهای درخت‌های پیشین تمرکز می‌کند. جمعِ درخت‌های ضعیف یک مدلِ قوی می‌سازد.
- گرادیان بوستینگ این «اصلاح» را با ریاضیاتِ گرادیان (جهتی که خطا را سریع‌ترین کاهش می‌دهد) انجام می‌دهد. XGBoost یک پیاده‌سازیِ سریع، منظم‌شده و صنعتیِ گرادیان بوستینگِ درختی است.

چرا XGBoost برای این مسئله؟ ویژگی‌های ما جدولی (سطر و ستونِ اعداد) هستند، به‌صورتِ غیرخطی با هم تعامل دارند و پُرنویزند. درخت‌های گرادیان‌بوست برای دقیقاً همین نوع داده، انتخابِ اثبات‌شده‌ی بهترین‌درکلاس‌اند — تعامل‌ها را خودکار می‌گیرند، با مقیاس‌های مختلط کنار می‌آیند و با منظم‌سازی در برابرِ بیش‌برازش مقاوم‌اند. آن را برای خروجیِ چندکلاسه تنظیم می‌کنیم (`multi:softprob` که برای هر یک از ۳ کلاس یک احتمال برمی‌گردانَد).

تنظیمِ ابرپارامتر با Optuna. یک مدل «ابرپارامتر» دارد (پیچ‌هایی مثلِ عمقِ درخت و نرخِ یادگیری). Optuna این‌ها را خودکار جست‌وجو می‌کند — ۴۰ آزمایش اجرا کرد، هر بار روی `train` آموزش و روی `val` امتیازدهی، و بهترین را نگه داشت. برنده: درخت‌های کم‌عمق (`max_depth=4`)، نرخِ یادگیریِ کند (`0.029`) و منظم‌سازیِ قوی — مدلی عمداً محتاط که به‌جای حفظ‌کردن، تعمیم می‌دهد. یک رگرسیونِ لجستیکِ ساده هم به‌عنوانِ خطمبنا برازش می‌دهیم؛ XGBoost از آن پیشی می‌گیرد (macro F1 0.375 در برابرِ 0.292)، که تأیید می‌کند پیچیدگیِ افزوده ارزشش را دارد.

گام ۵ — کالیبراسیونِ ایزوتونیک (صادق‌کردنِ احتمال‌ها)

یک مدل می‌تواند موارد را خوب رتبه‌بندی کند اما همچنان احتمال‌هایی بدمقیاس بدهد — مثلاً بگوید «۸۰٪» برای رویدادهایی که در واقع فقط ۶۰٪ مواقع رخ می‌دهند. کالیبراسیون *اعداد* را اصلاح می‌کند تا وقتی مدل می‌گوید ۷۰٪، رویداد واقعاً حدودِ ۷۰٪ مواقع رخ دهد.

رگرسیونِ ایزوتونیک یک روشِ کالیبراسیونِ منعطف و ناپارامتری است: هر نگاشتِ یکنواخت (همواره‌صعودی) از امتیازِ خام به فراوانیِ واقعی را یاد می‌گیرد. آن را فقط روی مجموعه‌ی اعتبارسنجی برازش می‌دهیم (هرگز روی آموزش یا آزمون — وگرنه تقلب می‌کند). نتیجه احتمال‌های صادق است، که اینجا مهم است چون تحلیل‌گر و رابطِ کاربری اعدادِ اطمینان را به کاربر نشان می‌دهند — آن اعداد باید همان معنایی را بدهند که می‌گویند.
کالیبراسیون را با امتیازِ Brier بررسی می‌کنیم (0.606؛ کمتر بهتر است).

گام ۶ — SHAP (توضیحِ هر پیش‌بینی)

مدلی که فقط می‌گوید «DOWN» یک جعبه‌ی سیاه است. SHAP آن را باز می‌کند. SHAP بر پایه‌ی مقادیرِ شپلی (Shapley) است، ایده‌ای از نظریه‌ی بازی‌ها: هر ویژگی را مثلِ یک «بازیکن» در یک تیم در نظر بگیر و اعتبار (یا تقصیرِ) پیش‌بینیِ نهایی را عادلانه میانِ آن‌ها تقسیم کن. برای هر پیش‌بینی، SHAP می‌گوید هر ویژگی نتیجه را چقدر بالا یا پایین هل داده است.

از `TreeExplainer` SHAP استفاده می‌کنیم (برای مدل‌های درختی سریع و دقیق) و سه محرکِ برتر را برای هر کوین در جدولِ `prediction_drivers` ذخیره می‌کنیم — تا رابطِ کاربری بتواند *چرا* را نشان دهد، مثلاً «بزرگ‌ترین دلایلِ DOWNبودن، روزِ هفته، ADX و سیگنالِ MACD بودند». (ویژگیِ هویتِ کوین از محرک‌ها کنار گذاشته می‌شود، پس دلایل، *وضعیتِ بازار* را توصیف می‌کنند نه صرفاً «این کوین BTC است».)

گام ۷ — ارزیابی: واقعاً چقدر خوب است؟

ROC-AUC — متریکِ سرتیتر، با توضیحِ کامل

ROC-AUC (سطحِ زیرِ منحنیِ مشخصه‌ی عملکردِ گیرنده) اندازه می‌گیرد که مدل چقدر خوب موارد مثبت را بالاتر از منفی‌ها رتبه‌بندی می‌کند. تصور کنید یک موردِ واقعیِ «UP» و یک موردِ واقعیِ «غیرِ-UP» را تصادفی انتخاب کنید: AUC احتمالِ آن است که مدل به موردِ واقعیِ UP امتیازِ بالاتری بدهد. آن را این‌گونه بخوانید:

- AUC = 0.5 ← مدل بهتر از پرتابِ سکه نیست (بدونِ مهارت).
- AUC = 1.0 ← رتبه‌بندیِ کامل.
- AUC = 0.578 (ما) ← یک برتریِ کوچک اما واقعی نسبت به تصادف — مدل *چیزی* یاد گرفته، فقط نه زیاد.

چرا AUC اینجا سرتیترِ منصفانه است؟ چون به یک آستانه‌ی تصمیمِ واحد وابسته نیست و فریبِ کلاس‌های نامتوازن را نمی‌خورد (FLAT رایج‌ترین نتیجه است). یک مدلِ ساده‌لوح که همیشه «FLAT» می‌گوید ممکن است دقیق به‌نظر برسد اما AUC حدودِ ۰٫۵ می‌گیرد؛ مدلِ ما 0.578 می‌گیرد، که دقیقاً در باندِ صادقانه و موردِانتظار برای پیش‌بینیِ جهتِ کوتاه‌مدتِ کریپتو از روی داده‌ی قیمت است. هر چیزی خیلی بالاتر از حدودِ ۰٫۶۵ اینجا، پرچمِ قرمزِ نشتِ داده است، نه مهارتِ واقعی.

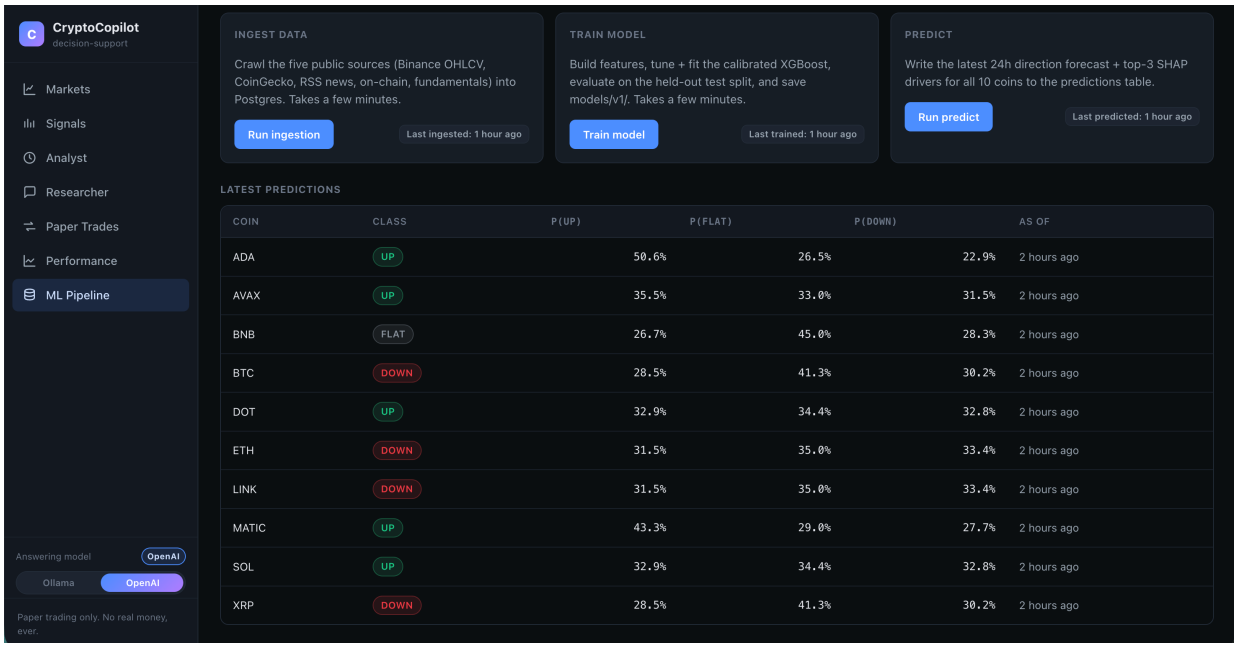
سایرِ متریک‌ها

متریک	مقدارِ ما	هدف	معنا
macro ROC-AUC	0.578	0.55–0.62	مهارتِ رتبه‌بندی (بالای ۰٫۵ = برتریِ واقعی). قبول.
multiclass Brier	0.606	≥ 0.65	دقتِ احتمال (کمتر بهتر است). قبول.
macro F1	0.375	≤ 0.40	توازنِ دقت و فراخوانی در کلاس‌ها. کمی کمتر — پایین را ببینید.

خطمبنای LogReg F1	0.292	—	خطمبنای ساده؛ XGBoost به‌روشنی از آن بهتر است.

چرا macro F1 برابرِ 0.375 است (و چرا این صادقانه است، نه یک باگ). سخت‌ترین کلاس UP است — پیش‌بینیِ یک رشدِ بیش از ۲٪ در ۲۴ ساعتِ آینده فقط از روی داده‌ی قیمت، در بازارِ آرامِ ۲۰۲۵–۲۶، نزدیکِ کفِ نویز است. تیم همه‌ی مظنون‌های معمول را بررسی و رد کرد: نشت نیست (AUC در باندِ موردِانتظار است؛ یک آزمونِ نشت قبول می‌شود)، قاعده‌ی تصمیم نیست (حتی یک قاعده‌ی بهینه با دیدِ پسینی هم در همان 0.375 سقف می‌خورَد)، و تنگ‌کردنِ هدف آن را بدتر کرد. محدودیتِ واقعی حجمِ داده است — حدودِ ۲ سال تاریخچه. تاریخچه‌ی بیشتر محتمل‌ترین اهرم برای عبور از ۰٫۴۰ است. این به‌عنوانِ نتیجه‌ی عمدی و محدودشده‌به‌داده گزارش می‌شود.

تصویر ۳ — سیگنالِ یک کوین. کلاسِ مدل (DOWN، ۳۰٫۲٪) و نوارِ احتمال، سپس سه محرکِ برترِ SHAP، سپس حکمِ قطعیِ تکنیکالِ ta4j با قواعدِ دقیقی که فعال شدند.



تصویر ۴ — صفحه‌ی ML Pipeline. سه کار (ingest ← train ← predict) را می‌توان به‌صورتِ درخواستی اجرا کرد؛ در پایین، تازه‌ترین احتمال‌های هر کوین برای UP / FLAT / DOWN.

5. سرویسِ بک‌اند (جاوا / Spring Boot)

بک‌اند «مغزِ» برنامه است. با Java 21 و Spring Boot نوشته شده، چارچوبِ استانداردِ صنعت برای ساختِ برنامه‌های سرورِیِ قابل‌اتکا (همه‌چیز را به هم وصل می‌کند، اتصالِ پایگاه‌داده را مدیریت می‌کند و REST API را سرو می‌کند). این یک مونولیتِ ماژولارِ واحد است با ماژول‌های داخلی برای داده، تحلیل تکنیکال، RAG، تحلیل‌گر و معاملات.

Spring AI — چیست و چگونه استفاده می‌کنیم

Spring AI کتابخانه‌ی رسمیِ Spring برای افزودنِ قابلیت‌های هوش مصنوعی به یک برنامه‌ی جاوا است. راهی تمیز و مستقل‌ازفراهم‌کننده می‌دهد تا یک مدلِ چت (LLM) و یک مدلِ embedding را فراخوانی کنیم و با یک پایگاه‌داده‌ی برداری صحبت کنیم. از آن برای دو چیز استفاده می‌کنیم: چتِ پژوهشگر (RAG) و خلاصه‌ی LLM-نوشته‌ی تحلیل‌گر. چون Spring AI فراهم‌کننده را پشتِ یک واسط پنهان می‌کند، می‌توانیم با تغییرِ پیکربندی میانِ Ollamaی محلی و OpenAI جابه‌جا شویم — کدِ برنامه تغییر نمی‌کند.

ta4j — اندیکاتورهای تحلیل تکنیکال در جاوا

ta4j («تحلیل تکنیکال برای جاوا») یک کتابخانه‌ی متن‌بازِ اندیکاتورهای معاملاتی است. بک‌اند از آن برای محاسبه‌ی حکمِ تکنیکالِ خودش مستقیماً از کندل‌های خامِ `ohlcv` استفاده می‌کند — کاملاً مستقل از مدلِ ML پایتون. اندیکاتورهای پیاده‌سازی‌شده:

اندیکاتور	چه چیزی را می‌سنجد	CryptoCopilot چگونه استفاده می‌کند
ابرِ ایچیموکو	روند، حمایت/مقاومت، مومنتوم (یک سیستمِ کامل)	محور: قیمت بالا/پایینِ ابر است؟ ابر صعودی است؟ (تنظیماتِ 52/26/9)
RSI (14)	مومنتوم: اشباعِ خرید (>۷۰) / فروش (<۳۰)	حکم را تعدیل می‌کند — خوانشِ اشباعِ فروش می‌تواند امتیازِ نزولی را نرم کند.
MACD (12,26,9)	روند و مومنتوم از اختلافِ میانگین‌های متحرک	هیستوگرامِ مثبتِ صعودی یک امتیازِ صعودی می‌افزاید.

بولینگر B (20,2)%	جای قیمت درونِ باندهای نوسانش	شرایطِ کشیده‌شده / بازگشت‌به‌میانگین را علامت می‌زند.
ATR	نوسان (میانگینِ محدوده‌ی واقعی)	زمینه‌ای برای بزرگیِ حرکت‌های اخیر.

این قواعد در یک امتیازِ عددی ترکیب می‌شوند که به یک جهت (BULLISH / NEUTRAL / BEARISH) و یک اطمینان (STRONG / MODERATE / WEAK) نگاشت می‌شود، به‌علاوه‌ی فهرستِ دقیقِ قواعدی که فعال شدند — پس حکم کاملاً شفاف و بازتولیدپذیر است. یک نکته‌ی مهم: «جابه‌جاییِ» ایچیموکو به‌شیوه‌ای ایمن در برابرِ نشت اعمال می‌شود که سمتِ پایتون را آینه می‌کند، پس اندیکاتور هرگز تصادفاً از کندل‌های آینده استفاده نمی‌کند.

کتابخانه‌های جاوای استفاده‌شده، و چرا

کتابخانه	هدف
Spring Boot (web, data-jpa)	چارچوبِ برنامه: نقاطِ REST، تزریقِ وابستگی، پیکربندی.
Spring Data JPA + Hibernate	نگاشتِ اشیای جاوا به جدول‌ها؛ `ddl-auto: validate` تطابقِ کد با اسکیما را بررسی می‌کند.
درایورِ JDBC PostgreSQL	اتصالِ جاوا به پایگاه‌داده‌ی Postgres.
Spring AI (+ pgvector + Ollama/OpenAI)	چتِ RAG و خلاصه‌ی تحلیل‌گر؛ ذخیره‌ی برداری.
ta4j	اندیکاتورهای تحلیل تکنیکال (حکمِ TA).
springdoc-openapi	تولیدِ خودکارِ مستنداتِ تعاملیِ API (Swagger UI).
JUnit 5 + Mockito	تست‌های خودکار (۷۰ تستِ آفلاین در بک‌اند).

کلِ API با Swagger UI (یک صفحه‌ی وبِ تعاملی که هر نقطه‌ی پایانی را فهرست می‌کند) خودمستند است که به‌صورتِ خودکار توسطِ springdoc تولید می‌شود.

Signals

The calibrated ML forecast (class · confidence · top SHAP drivers) beside the independent ta4j verdict, per coin.

BTC — DOWN 30.2%

UP 29% · FLAT 41% · DOWN 30%

model v1

TOP DRIVERS (SHAP)

#1	day_of_week	+0.094
#2	adx	−0.065
#3	macd_signal	−0.036

TECHNICAL ANALYSIS (TA4J)

NEUTRAL · WEAK · score −0.5

- Price below the Ichimoku cloud (−2.0)
- Bearish cloud: Senkou A below Senkou B (−0.5)
- MACD histogram positive and rising (+1.0)
- RSI oversold <30 (+1.0)

ETH — DOWN 33.4%

UP 32% · FLAT 35% · DOWN 33%

model v1

TOP DRIVERS (SHAP)

#1	adx	−0.107
#2	day_of_week	+0.096
#3	macd_hist	+0.095

TECHNICAL ANALYSIS (TA4J)

NEUTRAL · WEAK · score −1.5

- Price below the Ichimoku cloud (−2.0)
- Bearish cloud: Senkou A below Senkou B (−0.5)
- RSI oversold <30 (+1.0)

SOL — UP 32.9%

UP 33% · FLAT 34% · DOWN 33%

model v1

TOP DRIVERS (SHAP)

#1	day_of_week	−0.279
#2	atr_pct	+0.141
#3	adx	+0.100

TECHNICAL ANALYSIS (TA4J)

NEUTRAL · WEAK · score −1.5

- Price below the Ichimoku cloud (−2.0)
- Bearish cloud: Senkou A below Senkou B (−0.5)
- RSI oversold <30 (+1.0)

BNB — FLAT 45.8%

UP 27% · FLAT 45% · DOWN 28%

model v1

TOP DRIVERS (SHAP)

#1	atr_pct	−0.144
#2	day_of_week	+0.087
#3	ichimoku_cloud_thickness	+0.075

TECHNICAL ANALYSIS (TA4J)

NEUTRAL · WEAK · score −1.5

- Price below the Ichimoku cloud (−2.0)
- Bullish cloud: Senkou A above Senkou B (+0.5)

XRP — DOWN 30.2%

UP 29% · FLAT 41% · DOWN 30%

model v1

TOP DRIVERS (SHAP)

#1	day_of_week	+0.098
#2	adx	−0.076
#3	macd_hist	+0.070

TECHNICAL ANALYSIS (TA4J)

NEUTRAL · WEAK · score −1.5

- Price below the Ichimoku cloud (−2.0)
- Bearish cloud: Senkou A below Senkou B (−0.5)
- RSI oversold <30 (+1.0)

ADA — UP 50.6%

UP 51% · FLAT 26% · DOWN 23%

model v1

TOP DRIVERS (SHAP)

#1	day_of_week	−0.256
#2	atr_pct	+0.166
#3	sma90_ratio	+0.087

TECHNICAL ANALYSIS (TA4J)

NEUTRAL · WEAK · score −1.5

- Price below the Ichimoku cloud (−2.0)
- Bearish cloud: Senkou A below Senkou B (−0.5)
- RSI oversold <30 (+1.0)

AVAX — UP 35.5%

UP 36% · FLAT 33% · DOWN 31%

model v1

TOP DRIVERS (SHAP)

#1	day_of_week	−0.304
#2	atr_pct	+0.146
#3	adx	+0.099

TECHNICAL ANALYSIS (TA4J)

NEUTRAL · WEAK · score −1.5

- Price below the Ichimoku cloud (−2.0)
- Bearish cloud: Senkou A below Senkou B (−0.5)
- RSI oversold <30 (+1.0)

DOT — UP 32.9%

UP 33% · FLAT 34% · DOWN 33%

model v1

TOP DRIVERS (SHAP)

#1	day_of_week	−0.271
#2	atr_pct	+0.127
#3	adx	+0.084

TECHNICAL ANALYSIS (TA4J)

NEUTRAL · WEAK · score −1.5

- Price below the Ichimoku cloud (−2.0)
- Bearish cloud: Senkou A below Senkou B (−0.5)
- RSI oversold <30 (+1.0)

LINK — DOWN 33.4%

UP 32% · FLAT 35% · DOWN 33%

model v1

TOP DRIVERS (SHAP)

#1	day_of_week	+0.091
#2	adx	−0.081
#3	macd_hist	+0.062

TECHNICAL ANALYSIS (TA4J)

NEUTRAL · WEAK · score −1.5

- Price below the Ichimoku cloud (−2.0)
- Bearish cloud: Senkou A below Senkou B (−0.5)
- RSI oversold <30 (+1.0)

MATIC — UP 43.3%

UP 43% · FLAT 29% · DOWN 28%

model v1

TOP DRIVERS (SHAP)

#1	day_of_week	−0.238
#2	atr_pct	+0.144
#3	bb_bandwidth	+0.096

TECHNICAL ANALYSIS (TA4J)

NEUTRAL · WEAK · score −1.5

- Price below the Ichimoku cloud (−2.0)
- Bullish cloud: Senkou A above Senkou B (+0.5)

تصویر ۵ — صفحه‌ی Signals: برای هر ۱۰ کوین، کلاسِ ML و اطمینان، نوارِ احتمال، سه محرکِ برترِ SHAP و حکمِ تکنیکالِ ta4j در کنارِ هم.

6. لایه‌ی هوش مصنوعی — Embeddingها و RAG (پژوهشگر)

Embedding چیست؟ (از پایه)

کامپیوترها واژه‌ها را نمی‌فهمند؛ اعداد را می‌فهمند. Embedding راهی است برای تبدیلِ یک تکه متن به یک فهرستِ اعداد (یک «بردار») که معنای آن را در خود نگه می‌دارد. ویژگیِ جادویی: متن‌هایی با معنای مشابه بردارهای مشابه می‌گیرند، حتی اگر واژه‌های متفاوتی به‌کار ببرند. «سختیِ استخراجِ بیت‌کوین بالا رفت» و «تنظیمِ هش‌ریتِ BTC افزایش یافت» نزدیکِ هم می‌نشینند؛ «من پیتزا دوست دارم» دور می‌افتد.

هرِ embedding ما فهرستی از ۷۶۸ عدد است. برای یافتنِ متنِ مرتبط با یک پرسش، پرسش را embed می‌کنیم و سپس دنبالِ تکه‌های ذخیره‌شده‌ای می‌گردیم که بردارشان به آن نزدیک‌ترین است. «نزدیکی» با شباهتِ کسینوسی سنجیده می‌شود — در اصل، زاویه‌ی میانِ دو بردار؛ زاویه‌ی کوچک یعنی معنای مشابه.

RAG چیست، و چرا از آن استفاده کنیم؟

RAG مخففِ تولیدِ تقویت‌شده‌بابازیابی است. یک مدلِ زبانیِ بزرگ (LLM) به‌تنهایی با کمالِ میل چیزها را از خود می‌سازد («توهم می‌زند») و از داده‌ی خصوصی و به‌روزِ *شما* خبر ندارد. RAG هر دو مشکل را در سه گام حل می‌کند:

1. بازیابی — پرسشِ کاربر را embed و مرتبط‌ترین تکه‌های واقعی را از داده‌ی خودمان (اخبار، خلاصه‌های آن‌چین، فاندامنتال‌ها و پایگاه‌دانش) دریافت کن.
2. تقویت — آن تکه‌ها را به‌عنوانِ *تنها* مادهٔ مجاز در پرامپت بچسبان.
3. تولید — از LLM بخواه فقط با استفاده از همان تکه‌ها پاسخ دهد و با نشانه‌هایی مثلِ [1]، [2] به آن‌ها ارجاع دهد.

نتیجه، پاسخی است که در منابعِ واقعی ریشه دارد و ارجاع‌دار است، نه داستانِ بااعتمادبه‌نفس. پژوهشگرِ CryptoCopilot در این‌باره سخت‌گیر است: اگر منابع پاسخ را نداشته باشند، به‌جای حدس‌زدن با یک عبارتِ ثابت امتناع می‌کند، و همیشه از دادنِ توصیه‌ی معاملاتی خودداری می‌کند. در یک ارزیابیِ ۲۰ پرسشی به recall@8 = 0.90 رسید (برای ۹۰٪ پرسش‌ها، یک تکه‌ی درست در ۸ موردِ برترِ بازیابی‌شده بود) با نرخِ ارجاعِ ۱۰۰٪.

چرا pgvector؟

برای بازیابی بر اساسِ معنا، به پایگاه‌داده‌ای نیاز داریم که بتواند بردارها را ذخیره و نزدیک‌ترین‌ها را سریع پیدا کند. pgvector یک افزونه است که یک نوعِ ستونِ `vector` و جست‌وجوی شباهتِ سریع را به Postgres اضافه می‌کند — همان پایگاه‌داده‌ای که از پیش داریم. فایده‌ی کلیدی همین است: بدونِ پایگاه‌داده‌ی اضافی. embeddingها و داده‌ی رابطه‌ای‌مان در یک نمونه‌ی Postgres زندگی می‌کنند. Spring AI جدولِ `vector_store` را خودکار می‌سازد و مدیریت می‌کند (۷۶۸-بُعدی، با ایندکسِ HNSW برای جست‌وجوی سریعِ تقریبیِ نزدیک‌ترین‌همسایه با فاصله‌ی کسینوسی).

Ollama در برابرِ OpenAI — چرا هر دو وجود دارند (کلیدِ تعویض)

CryptoCopilot می‌تواند از دو فراهم‌کننده‌ی هوش مصنوعی استفاده کند، که با یک کلید در نوارِ کناری در زمانِ اجرا انتخاب می‌شوند:

- Ollama (پیش‌فرض) یک مدل را به‌صورتِ محلی روی کامپیوترِ خودتان اجرا می‌کند — مدلِ `llama3.2:3b`. این رایگان، خصوصی و بدونِ نیاز به کلیدِ API است. بده‌بستان این است که یک مدلِ محلیِ کوچک پاسخ‌های کوتاه‌تر و ساده‌تری می‌دهد.

- OpenAI (اختیاری) از مدلِ gpt-4o-mini در ابر استفاده می‌کند. پاسخ‌های به‌مراتب بهتر و روان‌تری می‌دهد، اما هزینه دارد و به کلیدِ API نیاز دارد.

داشتنِ هر دو، یک الگوی واقعیِ تولید را نشان می‌دهد: یک پیش‌فرضِ رایگان و کاملاً محلی که برای هرکسی کار می‌کند، به‌علاوه‌ی یک ارتقای یک‌کلیکی به یک مدلِ ابریِ قوی‌تر وقتی کیفیت مهم است. به‌لطفِ Spring AI، زدنِ این کلید فقط تعیین می‌کند کدام فراهم‌کننده پاسخ دهد — بقیه‌ی سامانه دست‌نخورده می‌مانَد.

Embeddingها: nomic-embed-text در برابرِ text-embedding-3-small

یک نکته‌ی ظریفِ مهم وجود دارد. کلیدِ بالا فقط مدلِ چت / خلاصه را تغییر می‌دهد. embeddingها همیشه روی مدلِ `Ollama` nomic-embed-text` می‌مانند (۷۶۸ عدد)، حتی وقتی چت به OpenAI سوییچ شود. چرا؟ چون کلِ ایندکسِ برداری با بردارهای ۷۶۸-بُعدی ساخته شده؛ تغییرِ مدلِ embedding، اندازه‌ی بردار را عوض می‌کند و یک ایندکس‌گذاریِ مجددِ کامل و کُند را بر همه‌ی تکه‌ها تحمیل می‌کند. ثابت‌نگه‌داشتنِ embedding یعنی کلیدِ تعویض فوری و رایگان است.

`text-embedding-3-small` مدلِ embeddingِ کوچک، سریع و کم‌هزینه‌ی OpenAI است (۱۵۳۶ عدد تولید می‌کند). نقشه‌ی نخستینِ پروژه بود و هنوز به‌عنوانِ گزینه‌ی بازگشت سیم‌کشی شده است. مدلِ embeddingِ بسیار خوب و کم‌هزینه‌ای است — اما چون پذیرشِ آن نیازمندِ ایندکس‌گذاریِ مجددِ پیکره در ۱۵۳۶ بُعد است، CryptoCopilot عمداً embeddingها را روی `nomic-embed-text`ِ رایگان و محلی نگه می‌دارد تا کلِ لایه‌ی بازیابی حدودِ ۰ یورو هزینه داشته باشد و هنگامِ تغییرِ فراهم‌کننده‌ی چت هرگز نیاز به بازسازی نباشد.

> به‌طورِ خلاصه: چت/خلاصه = Ollama llama3.2:3b یا OpenAI gpt-4o-mini (به انتخابِ شما، فوری).
> embedding/جست‌وجو = همیشه ۷۶۸، Ollama nomic-embed-text-بُعدی (پس هرگز نیازی به ایندکسِ مجدد نیست).

7. تجمیع‌کننده‌ی تحلیل‌گر (Analyst)

تحلیل‌گر بخشی است که چهار دیدگاه را در یک نظر ترکیب می‌کند. این بخش قطعی (deterministic) است — با ورودیِ یکسان همیشه همان حکم را تولید می‌کند — که آن را حسابرسی‌پذیر می‌سازد. هر ورودی روی یک مقیاسِ کوچکِ −۲ تا +۲ امتیاز می‌گیرد:

- ML — UP یا DOWN وقتی اطمینانِ کالیبره‌شده از آستانه عبور کند ±۲ می‌دهد (وگرنه ±۱)؛ FLAT صفر می‌دهد.
- تکنیکال (ta4j) — BULLISH/BEARISH × STRONG/MODERATE به ±۲ / ±۱ نگاشت می‌شود.
- سلامتِ فاندامنتال — IMPROVING / DETERIORATING مقدارِ ±۱ می‌دهد.
- احساساتِ خبری — مثبت / منفیِ خالص طیِ ۷ روزِ اخیر مقدارِ ±۱ می‌دهد.

چهار امتیاز در یک مجموعِ −۶ تا +۶ جمع می‌شوند، که تبدیل می‌شود به:

- جهت — LEAN_BULLISH (≥ +۳)، LEAN_BEARISH (≤ −۳)، CONFLICTED (ورودی‌ها در علامت اختلاف دارند)، یا NEUTRAL.
- قطعیت — HIGH (|مجموع| ≥ ۴)، MEDIUM (۲–۳)، یا LOW.
- امتیازِ توافق — اینکه چهار ورودی چقدر هم‌نظرند (۱ = توافقِ کامل).

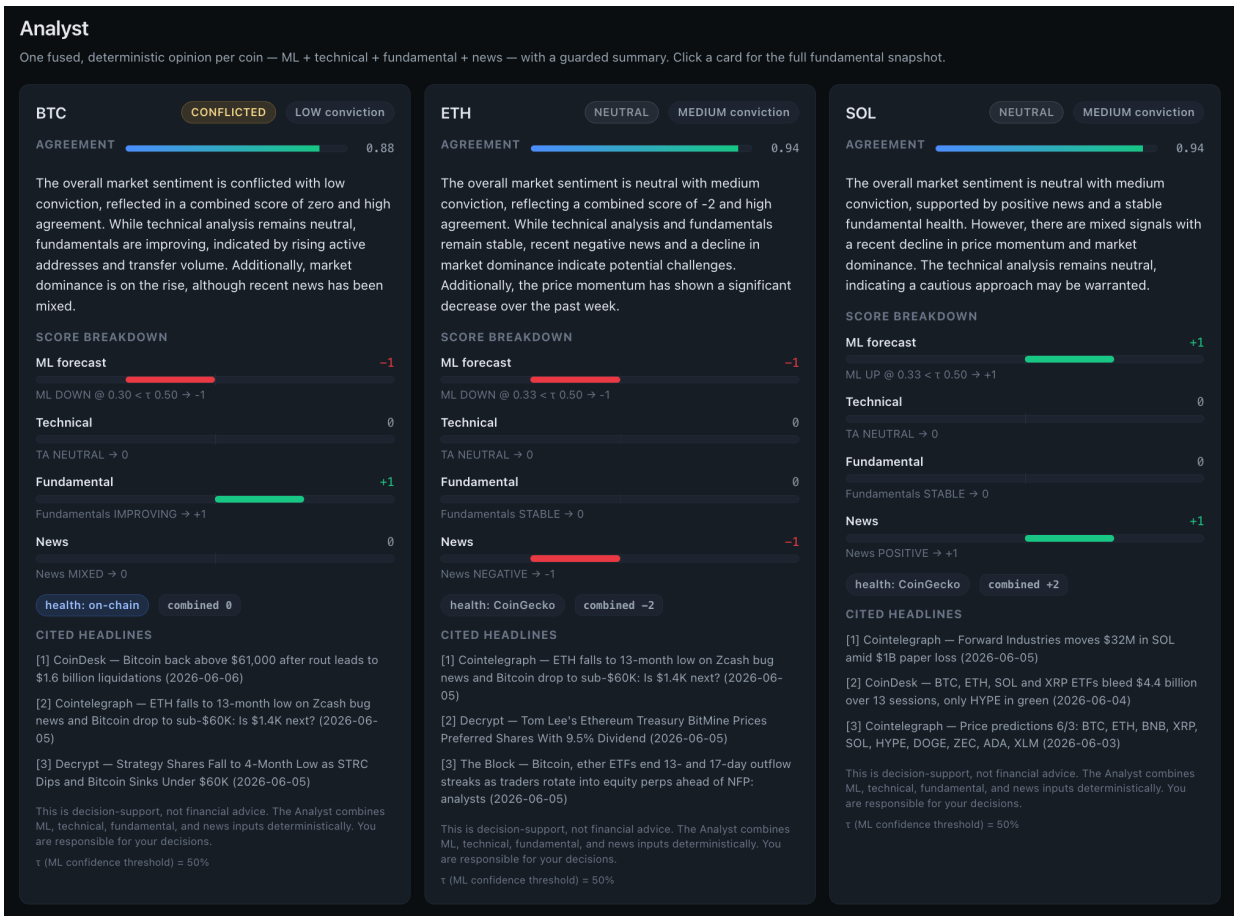
سلامتِ فاندامنتالِ دو-لایه

«سلامتِ» فاندامنتال از بهترین داده‌ی موجود محاسبه می‌شود و منبع به‌صراحت به‌صورتِ `healthSource` گزارش می‌شود:

- لایه ۱ — آن‌چین (سری زمانیِ روزانه‌ی واقعی): میانگین‌های ۷-روزه‌ی صعودیِ آدرس‌های فعال و حجمِ انتقال ← IMPROVING. در عمل فقط BTC این سری زمانیِ روزانه‌ی غنی را دارد.
- لایه ۲ — CoinGecko: برای بقیه‌ی کوین‌ها، یک قاعده روی مومنتومِ ۷-روزه، فعالیتِ توسعه‌دهنده و تغییرِ ۲۴-ساعته‌ی ارزشِ بازار.
- لایه ۳ — نامعلوم: اگر هیچ‌کدام موجود نباشد، کوین صرفاً UNKNOWN علامت می‌خورد — هرگز حدس زده نمی‌شود.

نگهبانِ توهم (Hallucination Guard)

خلاصه‌ی خوانای تحلیل‌گر توسطِ LLM عبارت‌پردازی می‌شود، اما هر عدد در آن خلاصه باید از پیش در ورودی‌ها وجود داشته باشد. یک نگهبانِ توهم این را بررسی می‌کند: اگر مدل عددی از خود بسازد، یا LLM در دسترس نباشد، سامانه به یک قالبِ قطعی ساخته‌شده از اعدادِ واقعی برمی‌گردد. پس تحلیل‌گر حتی با LLM خاموش قابل‌اتکاست — صرفاً همان واقعیت‌ها را ساده‌تر بیان می‌کند.



تصویر ۶ — صفحه‌ی Analyst برای سه کوین. هرکدام جهت (مثلاً CONFLICTED / NEUTRAL)، امتیازِ توافق، یک خلاصه‌ی به‌زبانِ ساده، تفکیکِ امتیاز در ML / تکنیکال / فاندامنتال / اخبار، منبعِ سلامت و تیترهای ارجاع‌دار را نشان می‌دهد.

8. موتورِ معامله‌ی آزمایشی (Paper Trading)

«معامله‌ی آزمایشی» یعنی تمرین با پولِ فرضی — همان سازوکارِ معامله‌ی واقعی، اما هرگز چیزی در خطر نیست. این یک قانونِ سختِ پروژه است: هرگز پولِ واقعی؛ فقط خرید (long-only)؛ بدونِ فروشِ استقراضی؛ بدونِ اهرم. موتور به کاربر اجازه می‌دهد سفارش بگذارد و سپس حساب را ردیابی می‌کند.

یک «پُرشدن (fill)» چگونه کار می‌کند

منطقِ «پُرشدن» (اینکه یک سفارش چگونه به یک معامله‌ی کامل تبدیل می‌شود) یک‌بار نوشته و با بک‌تست به اشتراک گذاشته شده، پس شبیه‌سازی و رفتارِ زنده هماهنگ‌اند:

- سفارشِ MARKET — در قیمتِ بازگشاییِ کندلِ ۱-ساعته‌ی بعدی پُر می‌شود، که با یک لغزشِ (slippage) واقع‌گرایانه‌ی ۰٫۰۵٪ جابه‌جا شده (خریدار کمی بیشتر می‌پردازد، فروشنده کمی کمتر می‌گیرد).
- سفارشِ LIMIT — فقط اگر محدوده‌ی قیمتِ یک کندلِ بعدی واقعاً به قیمتِ حدی برسد پُر می‌شود؛ در غیرِ این صورت PENDING می‌مانَد.
- یک کارمزدِ تیکرِ ۰٫۱٪ روی هر پُرشدن گرفته می‌شود (درست مثلِ یک صرافیِ واقعی).

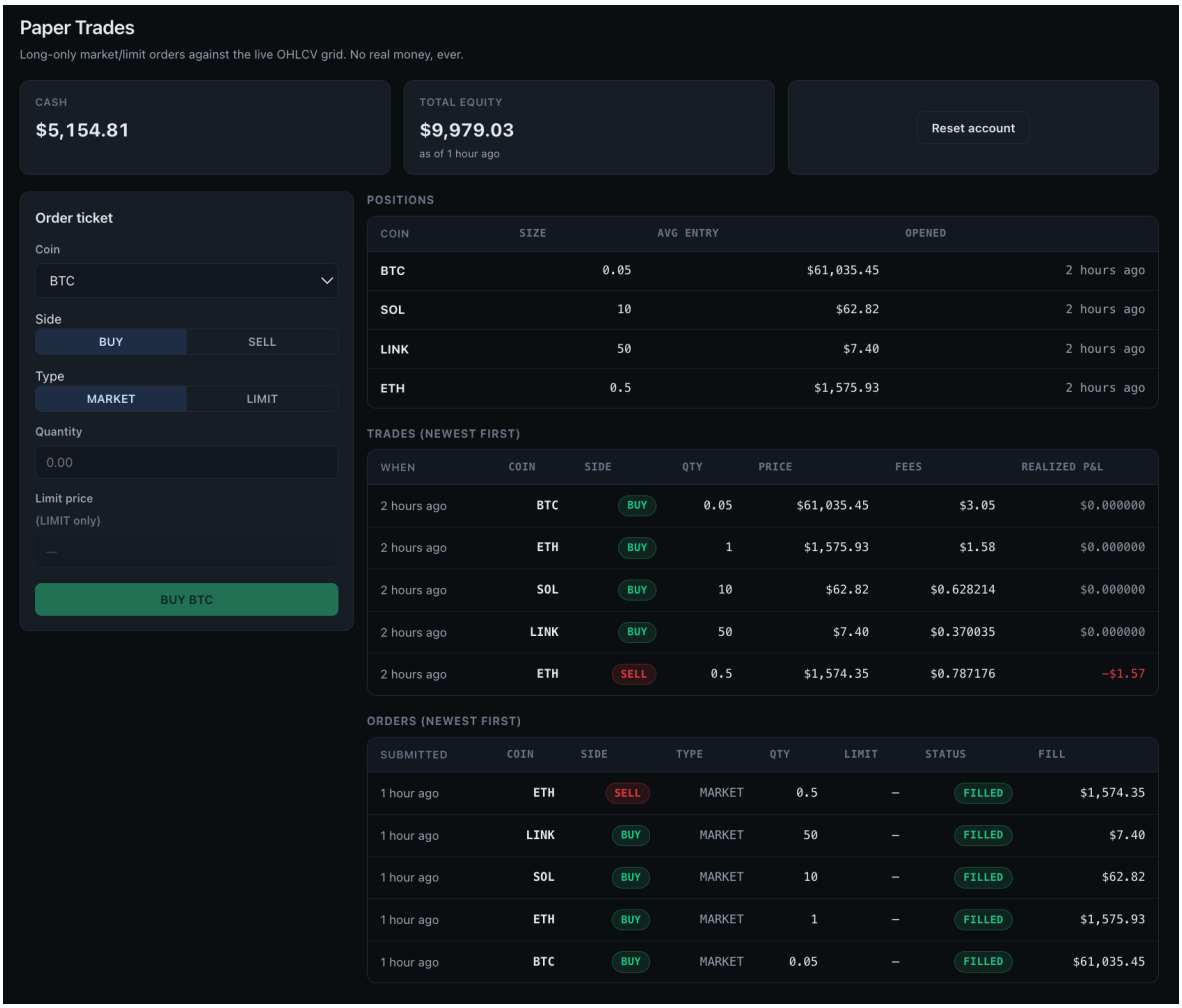
موتور چهار جدول را می‌نویسد — `orders`، `trades`، `positions` و `account_state` — و پس از هر پُرشدن یک تصویرِ تمیز از سرمایه می‌گیرد، پس منحنیِ سرمایه هرگز یک معامله‌ی نیمه‌اعمال‌شده را ثبت نمی‌کند. قواعد را نیز اعمال می‌کند: فروشِ بیش از آنچه داری رد می‌شود (فقط خرید).

متریک‌های عملکرد

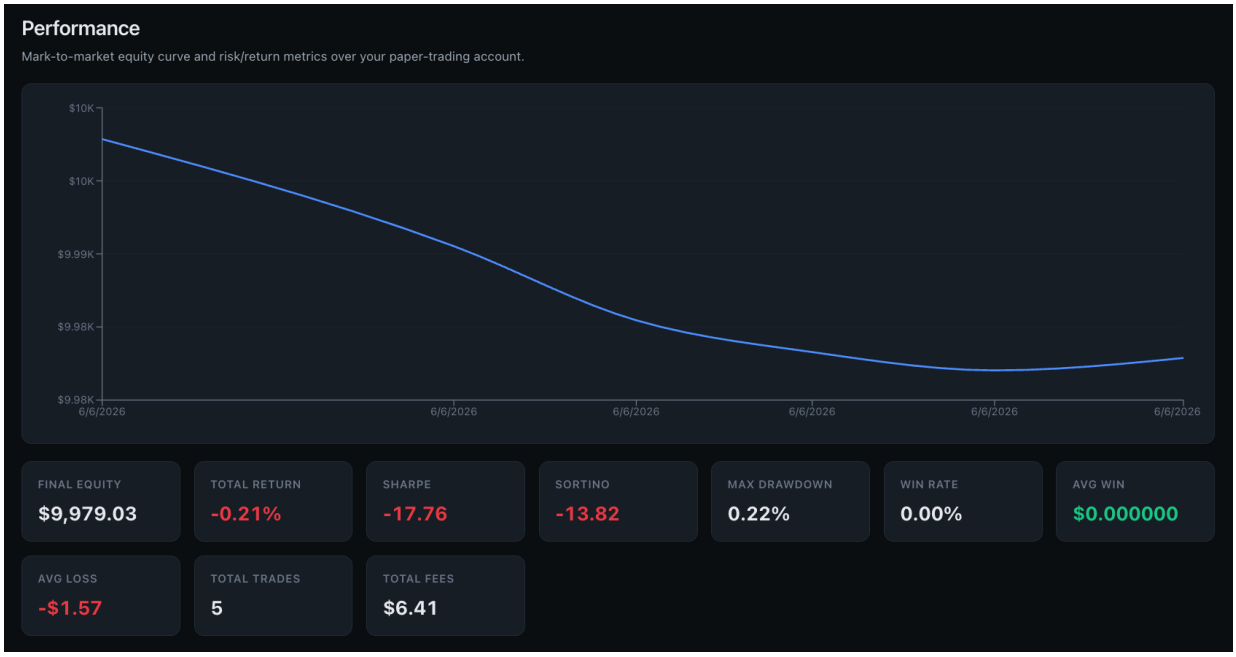
صفحه‌ی Performance حساب را به بازار نشانه‌گذاری می‌کند و متریک‌های استانداردِ ریسک/بازده را روی منحنیِ سرمایه محاسبه می‌کند: نسبت‌های شارپ و سورتینو (بازده به‌ازای هر واحدِ ریسک)، بیشینه‌ی افتِ سرمایه (بدترین سقوطِ قله‌تادره)، نرخِ برد، میانگینِ برد/باخت، کلِ معاملات و کلِ کارمزدها. (هرکدام در واژه‌نامه توضیح داده شده‌اند.)

بک‌تست — یک نتیجه‌ی صادقانه

یک بک‌تست یک استراتژی را روی تاریخِ واقعی بازپخش می‌کند. استراتژیِ پیش‌فرضِ مشخصات («ML-تأییدشده-با-TA») معامله می‌کند، به یک دلیلِ صادقانه: سرویسِ ML فقط آخرین پیش‌بینیِ هر کوین را ذخیره می‌کند، نه یک سری تاریخی، پس هیچ سیگنالِ ML گذشته‌ای برای راندنِ قاعده کندل‌به‌کندل وجود ندارد — و در این بازارِ آرام هیچ کوینی یک «UP» بااطمینان نیست. یک پروکسیِ فقط-TAی بازسازی‌پذیر *واقعاً* معامله می‌کند (۲۰۶ معامله) و در شارپِ −۱٫۲۰ می‌نشیند — یک نتیجه‌ی صادقانه‌ی منفی، چون کارمزد به‌علاوه‌ی بازارِ پُرنوسان یک استراتژیِ ساده‌ی روندی را می‌ساید. هدفِ این مرحله یک موتورِ معاملاتیِ درست و خوب‌آزموده است، نه یک استراتژیِ سودده.



تصویر ۷ — صفحه‌ی Paper Trades: نقدینگی و ارزشِ کل، تیکتِ سفارش (BUY/SELL · MARKET/LIMIT) و جدول‌های positions / trades / orders.



تصویر ۸ — صفحه‌ی Performance: منحنیِ سرمایه‌ی نشانه‌گذاری‌شده‌به‌بازار و متریک‌های ریسک/بازده (شارپ، سورتینو، بیشینه‌ی افت، نرخِ برد، کارمزدها).

9. فرانت‌اند (React)

فرانت‌اند وب‌سایتی است که با آن تعامل می‌کنید. یک کلاینتِ نازک است: هیچ منطقِ کسب‌وکاری از خود ندارد — صرفاً آنچه بک‌اند برمی‌گردانَد را نمایش می‌دهد و سفارش‌های آزمایشی را ثبت می‌کند.

انتخاب‌های فناوری

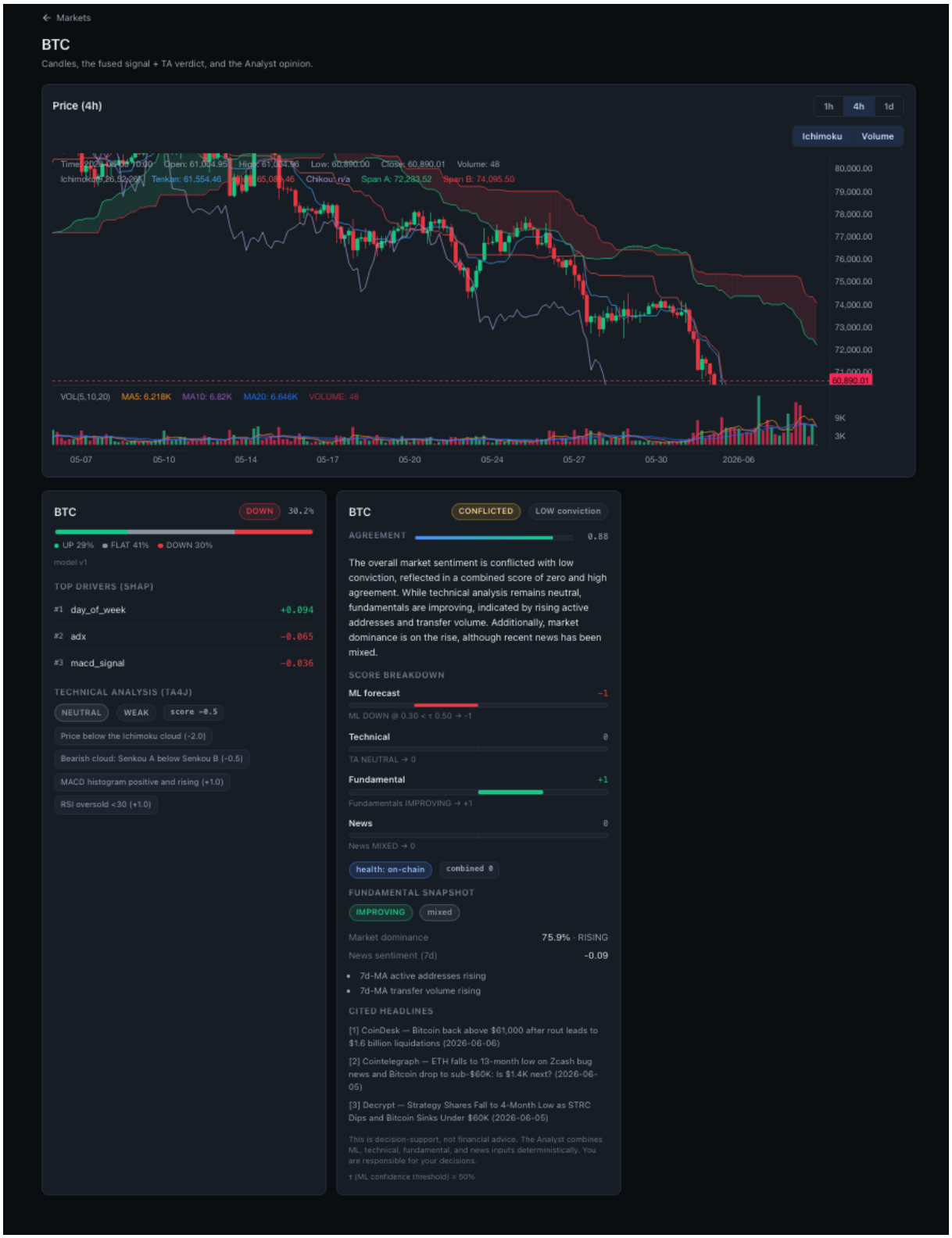
- React — محبوب‌ترین کتابخانه برای ساختِ رابطهای کاربریِ تعاملی از اجزای قابل‌استفاده‌ی مجدد.
- Vite — یک ابزارِ ساخت و سرورِ توسعه‌ی بسیار سریع.
- TypeScript — جاوااسکریپت با نوع (type)، که یک دسته‌ی کامل از باگ‌ها را پیش از اجرای کد می‌گیرد و شکلِ داده‌های بک‌اند را دقیقاً آینه می‌کند.
- nginx — یک وب‌سرورِ سریع که برنامه‌ی ساخته‌شده را سرو می‌کند و `/api` را به بک‌اند پراکسیِ معکوس می‌کند، پس مرورگر همیشه فقط با یک مبدأ صحبت می‌کند — یعنی هیچ پیکربندیِ CORS لازم نیست.

نمودارها

دو کتابخانه‌ی نموداری استفاده می‌شود: klinecharts نمودارهای شمعیِ حرفه‌ای (با پوششِ ایچیموکو و کلیدِ ۱d/۴h/۱h) را در صفحه‌ی جزئیاتِ کوین می‌کشد، و Recharts منحنیِ سرمایه را در صفحه‌ی Performance می‌کشد. (پروژه نخست برای شمع‌ها از TradingView Lightweight Charts استفاده می‌کرد و بعداً برای پوشش‌های اندیکاتورِ داخلیِ غنی‌ترش به klinecharts تغییر داد.)

صفحه‌ها

برنامه شش صفحه‌ی اصلی دارد — Researcher، Analyst، Signals، Markets (چت)، Paper Trades، Performance — به‌علاوه‌ی یک صفحه‌ی جزئیاتِ کوین و یک صفحه‌ی ML Pipeline (که می‌توانید ingest / train / predict را اجرا و تماشا کنید). یک سلبِ مسئولیتِ همیشگیِ «ابزارِ کمک‌به‌تصمیم، نه توصیه‌ی مالی» در همه‌ی صفحه‌ها هست.



تصویر ۹ — صفحه‌ی جزئیاتِ کوین: یک نمودارِ شمعیِ klinecharts با پوششِ ایچیموکو، در کنارِ سیگنالِ درون‌خطیِ ML و نظرِ کاملِ تحلیل‌گر برای آن کوین.

10. خطِ لوله‌ی درخواستی: ML API (پایتون / FastAPI)

در ابتدا کارهای ML فقط طبقِ زمان‌بندی اجرا می‌شدند. کانتینرِ ml-api یک لایه‌ی نازکِ وبِ FastAPI روی *همان* کدِ ingest / train / predict اضافه می‌کند، پس آن کارها می‌توانند به‌صورتِ درخواستی اجرا شوند — از بک‌اند (`/api/ml/*`) و صفحه‌ی ML Pipeline در رابطِ کاربری. این‌ها به‌صورتِ کارهای پس‌زمینه اجرا می‌شوند که می‌توانید وضعیتشان را poll کنید، و چون هم زمان‌بند و هم API همان پوشه‌ی مدل را به اشتراک می‌گذارند، مدلی که از رابطِ کاربری آموزش داده شود دقیقاً همانی است که پیش‌بینیِ بعدی استفاده می‌کند. FastAPI انتخاب شد چون راهِ استاندارد و سبک برای عرضه‌ی توابعِ پایتون به‌عنوانِ یک ِAPI سریعِ وب است، با مستنداتِ خودکارِ خودش.

11. واژه‌نامه — مفاهیمِ کلیدی به‌زبانِ ساده

هر اصطلاحِ مهمِ این پروژه، به‌اختصار تعریف شده.

کریپتو و بازار

- غیرمتمرکز بودن — هیچ مرجعِ واحدی (مثلِ بانکِ مرکزی) در کنترل نیست؛ شبکه را بسیاری از مشارکت‌کنندگانِ مستقل می‌گردانند.
- اجماع (PoW / PoS) — اینکه یک شبکه‌ی غیرمتمرکز چگونه بر سرِ حقیقت توافق می‌کند. اثباتِ کار (بیت‌کوین) انرژی خرج می‌کند؛ اثباتِ سهام (اتریوم، سولانا...) کوین‌ها را وثیقه می‌گذارد.
- توکنومیکس — اقتصادِ یک کوین: کلِ عرضه، نحوه‌ی ایجادِ کوینِ جدید و انگیزه‌ها.
- ارزشِ بازار (Market cap) — قیمت × عرضه‌ی در گردش؛ معیاری از اندازه‌ی کل.
- متریکِ آن‌چین — فعالیتی که مستقیماً روی بلاکچین ثبت می‌شود (مثلِ آدرس‌های فعال، حجمِ انتقال).
- OHLCV / کندل — قیمتِ بازگشایی، بیشترین، کمترین، بسته‌شدن و حجم برای یک بازه‌ی زمانی.

یادگیری ماشین

- ویژگی (Feature) — یک عددِ ورودی که وضعیتِ کنونی را توصیف می‌کند (مثلِ RSI، بازده‌ِ اخیر).
- دسته‌بندی (Classification) — پیش‌بینیِ یک رده (اینجا: UP / FLAT / DOWN).
- بیش‌برازش (Overfitting) — حفظ‌کردنِ داده‌ی آموزش به‌جای یادگیریِ الگوی عمومی؛ منظم‌سازی با آن می‌جنگد.
- گرادیان بوستینگ / XGBoost — درخت‌های تصمیمِ کوچکِ بسیار، ساخته‌شده پشتِ‌سرِ هم، هرکدام در حالِ اصلاحِ خطای قبلی.
- کالیبراسیون / ایزوتونیک — تنظیمِ احتمال‌ها تا «۷۰٪» واقعاً یعنی ۷۰٪.
- SHAP / مقدارِ شپلی — تقسیمِ عادلانه‌ی اعتبار میانِ ویژگی‌ها، که هر پیش‌بینی را توضیح می‌دهد.
- ROC-AUC — مهارتِ رتبه‌بندی (۰٫۵ = تصادفی، ۱٫۰ = کامل). Brier — دقتِ احتمال (کمتر بهتر). F1 — توازنِ دقت و فراخوانی.
- نشت / قرنطینه (Leakage / embargo) — استفاده‌ی تصادفی از اطلاعاتِ آینده؛ یک فاصله‌ی زمانی میانِ بُرش‌ها از آن جلوگیری می‌کند.

هوش مصنوعی / RAG

- LLM — مدلِ زبانیِ بزرگ، همان نوع هوش مصنوعی که متن تولید می‌کند.
- Embedding / بردار — متنی که به فهرستی از اعداد تبدیل شده که معنا را در خود دارد.
- شباهتِ کسینوسی — اینکه دو بردار (معنا) چقدر نزدیک‌اند.

- RAG — تولیدِ تقویت‌شده‌بابازیابی: منابعِ واقعی را بگیر، سپس فقط با آن‌ها پاسخ بده.
- توهم (Hallucination) — وقتی یک LLM با اطمینان چیزی نادرست می‌گوید؛ ریشه‌داری + نگهبان اینجا از آن جلوگیری می‌کنند.
- احساساتِ VADER — ابزاری سریع و قانون‌محور که متن را مثبت / منفی / خنثی امتیاز می‌دهد، برای برچسب‌زنیِ اخبار.

معاملات

- معامله‌ی آزمایشی — تمرینِ معامله با پولِ فرضی.
- لغزش (Slippage) — اختلافِ قیمتِ کوچک میانِ پُرشدنِ موردِانتظار و واقعی.
- کارمزدِ تیکر — کارمزدِ صرافی که هنگامِ پُرشدنِ فوریِ سفارش پرداخت می‌شود.
- شارپ / سورتینو — بازده به‌ازای هر واحدِ ریسک (سورتینو فقط ریسکِ نزولی را می‌شمارد).
- افتِ سرمایه (Drawdown) — اینکه حساب چقدر از قله‌ی پیشینش پایین آمده.

12. اجرا و نتایجِ صادقانه

چگونه کلِ سامانه را اجرا کنیم

یک فرمان سامانه را بالا می‌آورد و آن را با داده پُر می‌کند:

```
cp -n .env.example .env      # add the two free API keys
make demo                    # up + ingest + train + predict + reindex + seed trades
```

سپس http://localhost:3000 (برنامه)، :8080/swagger-ui.html (مستنداتِ API) و :8000/docs (ML API) را باز کنید. چتِ پژوهشگر و خلاصه‌ی LLM-نوشته‌ی تحلیل‌گر از یک Ollamaی محلی استفاده می‌کنند؛ با Ollamaی خاموش، چت به‌صورتِ تمیز امتناع می‌کند و تحلیل‌گر از قالبِ قطعیِ خود استفاده می‌کند — هر صفحه‌ی دیگری در هر دو حالت کاملاً پُر است.

نتایجِ صادقانه، در یک جدول

لایه	نتیجه	خوانشِ صادقانه
جهتِ ML	macro F1 0.375 · AUC 0.578 · Brier 0.606	یک برتریِ کوچک اما واقعی؛ AUC و Brier قبول؛ F1 سقفِ محدودشده‌به‌داده (~۲ سال).
بازیابیِ RAG	recall@8 0.90 · ۱۰۰٪ ارجاع	بازیابیِ ریشه‌دارِ قوی؛ از پرسش‌های خارج‌ازپیکره و توصیه‌ی معاملاتی امتناع می‌کند. ≈ ۰ یورو (محلی).
بک‌تستِ معامله	پیش‌فرض ۰ معامله · پروکسیِ TA شارپ −۱٫۲۰	موتور درست و آزموده است؛ استراتژی در این رژیم صادقانه سودده نیست.

این پروژه چه چیزی را نشان می‌دهد: یک معماریِ چندزبانه‌ی تمیز (دو زبان که از طریقِ یک پایگاه‌داده همکاری می‌کنند)، یک خطِ لوله‌ی_ML کامل، کالیبره‌شده و قابل‌توضیح، یک بک‌اندِ_Spring Boot تولیدمحور با تحلیل تکنیکال و یک چتِ_RAG ریشه‌دار، یک تحلیل‌گرِ تجمیعِ تصمیمِ قطعی با نگهبانِ توهم، یک موتورِ معامله‌ی آزمایشیِ ایمن، و یک فرانت‌اندِ ری‌اکتِ صیقلی و نوع‌دار — همه بسته‌بندی‌شده با داکر، تست‌ها و یکپارچه‌سازیِ پیوسته (CI).

> ⚠️ یادآوریِ پایانی: CryptoCopilot یک پروژه‌ی شخصیِ یادگیری و نمونه‌کار است. ابزارِ کمک‌به‌تصمیم است، نه توصیه‌ی مالی. فقط معامله‌ی آزمایشی — هرگز پولِ واقعی.