**نقشه راه و ساختار کلی استراتژی معاملات فیوچرز مبتنی بر یادگیری ماشینی با بهبود مستمر**

**1. طراحی و پیادهسازی معماری کلی سیستم**

* **ورودی دادهها:** دریافت دادههای بازار در لحظه شامل تیک دیتا، دفتر سفارش (Limit Order Book)، اخبار و احساسات بازار
* **پیشپردازش و Feature Store:** نرمالسازی دادهها، استخراج ویژگیها (مانند اندیکاتورها، imbalanceهای دفتر سفارش، نوسان) و ذخیره آنها در سیستم Feature Store با سرعت پایین
* **ماژول تشخیص تغییرات لحظهای و روندهای نوظهور:** الگوریتمهای حساس و دقیق (مانند CUSUM، BOCPD، Isolation Forest، LSTM- یا Transformer-based anomaly detection)
* **مجموعه مدلهای یادگیری ماشینی:**
  + مدلهای supervised مانند LSTM، Transformer، GNN برای پیشبینی
  + مدلهای یادگیری تقویتی محافظهکار (Offline/Conservative RL) مانند PPO یا SAC برای بهبود مستمر سیاست معاملاتی و تصمیمگیری
  + مدلهای قواعدی مالی محافظهکار برای مدیریت ریسک پایه
* **لایه ترکیب و تصمیمگیری (Ensembling):** استفاده از meta-learner (مانند stacking، weighted voting، Kalman filter) برای بهینهسازی سیگنالها و ایجاد تصمیمات کمریسک
* **اجرای سفارش:** مدیریت اندازه موقعیت (position sizing) با هدف نوسانگیری کنترل شده، مدیریت لوریج، و هماهنگی با مکان اجرای معاملات
* **مانیتورینگ و هشدار:** داشبوردهای realtime، سیستمهای هشدار (Slack، ایمیل، SMS) برای ریسکهای بحرانی، ناهنجاریها و افت عملکرد مدل
* **خط بازآموزی و آزمایش:** خط تولید بازآموزی اتوماتیک مبتنی بر triggerهای مشخص (مانند افت Sharpe، افزایش error) و تست سایهای (shadow testing) قبل از استقرار مدلهای تازه

**2. فازهای کلیدی توسعه و پیادهسازی (Roadmap اجرایی)**

| **فاز** | **اهداف کلیدی** | **توضیح مختصر** |
| --- | --- | --- |
| فاز 0: زیرساخت | راهاندازی pipeline داده realtime (Kafka)، Feature Store، ذخیره تاریخچه دیتا | زیرساخت برای جمعآوری و مدیریت دادهها |
| فاز 1: استراتژی پایه | پیادهسازی استراتژی قاعدهای (rule-based) و backtesting event-driven | تست اولیه و ایجاد اسکلت اصلی سیستم |
| فاز 2: تشخیص تغییرات و روند | پیادهسازی الگوریتمهای تشخیص مفهوم Drift و anomaly detection | واکنش سریع و دقیق به تغییرات بازار |
| فاز 3: توسعه مدلها | ایجاد چند مدل یادگیری ماشینی (LSTM، Transformer، GNN)، مدلهای یادگیری تقویتی محافظهکار | توسعه بخش پیشبینی و تصمیمگیری |
| فاز 4: ترکیب و مدیریت ریسک | اضافه کردن لایه ensemble و مدیریت ریسک خودکار (volatility targeting، stop-loss، kill switch) | کاهش ریسک و افزایش دقت تصمیمگیری |
| فاز 5: تست نهایی و اجرا | تست walk-forward، Monte Carlo، شبیهسازی بحران و shadow deployment | حصول اطمینان از عملکرد واقعی |
| فاز 6: بهبود مستمر | بازآموزی دورهای، بهبود الگوریتمها، افزودن مدلهای جدید، پایش مداوم عملکرد | حفظ و بهبود عملکرد مدل در زمان |

**3. نکات مهم در طراحی هر بخش**

* **تشخیص تغییر:** تفکیک دو سطح حساس و دقیق برای شناسایی تغییرات ناگهانی و تحلیلهای جزئیتر برای بازآموزی هدفمند
* **یادگیری تقویتی:** استفاده از ترکیب offline RL و مدلهای قواعدی جهت کنترل ریسک و پیشگیری از overfitting و نوسانات شدید
* **چارچوب تست:** event-driven backtesting با شبیهسازی سفارشها، ارزیابی rolling و تست استرس برای انعطافپذیری استراتژی در شرایط مختلف
* **مدیریت ریسک:** سطوح متعدد ریسک از اندازه موقعیت تا محدودیت اکسپوژر کل، همراه با هشدار و kill-switch در شرایط بحرانی
* **بازآموزی مدل:** سیستم trigger-based مبتنی بر معیارهای مالی و دقت مدل، با قابلیت warm-start و حفظ پایستگی مدل
* **ادغام مدلها:** وزندهی پویا و قواعد محافظهکارانه برای جلوگیری از تصمیمات متناقض و پرریسک

**4. موارد فنی و پلتفرم پیشنهادی**

* زبان برنامهنویسی اصلی: Python (کتابخانههای TensorFlow, PyTorch, Stable-Baselines3, scikit-learn)
* زیرساخت داده: Kafka، Feature Store (Feast یا داخلی)، ذخیرهسازی دادههای tick و عمق بلاک (LOB)
* ابزارهای تست: Backtrader، Zipline، MLOps framework برای اتوماسیون چرخه مدلها
* مراجع داده: APIهای لحظهای (Binance، Alpha Vantage) و دادههای تاریخی با کیفیت بالا
* سامانه هشدار: Slack، PagerDuty، ایمیل، SMS با داشبوردهای realtime
* ذخیرهسازی مدل و آزمایش سایهای قبل از استقرار نهایی

این نقشه راه و ساختار کلی، یک چشمانداز عملی و بهروز برای طراحی، توسعه و اجرای استراتژیهای معاملات فیوچرز مبتنی بر یادگیری ماشین را فراهم میکند و میتواند به عنوان راهنمای گامبهگام پروژه شما به کار رود. اگر بخواهید، میتوانم این ساختار را به چکلیست اجرایی، نمودار معماری یا یک فایل ارائه تخصصی تبدیل کنم و یا در جزئیات هر بخش به شما کمک کنم.

لطفاً اگر اطلاعات فنی خاصی درباره دادهها و زیرساخت خود دارید، بفرمایید تا نقشه راه را متناسب با شرایط دقیق شما بهینه کنم.

برای نزدیک کردن استراتژی پیشرفته معاملات فیوچرز مبتنی بر یادگیری ماشینی شما به مرحله واقعیت و قابل درک بودن، باید موارد زیر را به صورت عملی، ملموس و قابل فهم پیادهسازی کنیم:

۱. تعریف دقیق و ساده اجزای کلیدی استراتژی

* دادهها: مشخص کنید دقیقاً چه سطح دادههایی دارید (تیک دیتا، L1 یا L2، عمق کتاب سفارش) و منابع دریافت آنها چه هستند.
* مدلها: تعداد و نوع مدلهای یادگیری ماشینی (مثلاً ۲ مدل LSTM و ۱ مدل RL محافظهکار) را تعیین کنید.
* تصمیمگیری: یک روش ساده برای ترکیب خروجی مدلها تعیین کنید، مثلاً وزندهی خطی بر اساس دقت ماژولها یا یک meta-learner ساده.

۲. تهیه نمونه کدهای کاربردی و قابل اجرا برای بخشهای اساسی

* کد نمونه برای پیشپردازش داده و ساخت فیچرهای اصلی.
* کد ساده مدل پیشبینی (مثلاً یک LSTM کوتاهمدت) برای شفافیت روند.
* نمونه الگوریتم یادگیری تقویتی محافظهکار مانند PPO با دادههای مصنوعی برای تجربه عملی.
* کد نمونه ترکیب خروجی مدلها (به صورت میانگین وزنی یا stacking ساده).

۳. اجرای آزمایشی (Pilot) با دادههای کوچک و قابل کنترل

* راهاندازی یک محیط آزمایشی ساده با دادههای تاریخی چند هفته.
* استفاده از backtesting ایونتدرایو و گزارش دقیق metrikهای مالی (مثل sharpe, drawdown).
* شبیهسازی تغییرات ناگهانی با دادههای مصنوعی ساده برای دیده شدن واکنش سیستم.

۴. مستندسازی ساده و قابل فهم برای هر بخش

* شرح مرحله به مرحله نحوه کار سیستم و اجزای آن به زبان ساده و با نمودار.
* تعریف ورودی، خروجی و نقش هر ماژول با مثال.
* فلوچارت کلی مسیر دادهها و تصمیمات.

۵. نمایش داشبورد و گزارشهای قابل فهم

* ساخت یک داشبورد ساده جهت نمایش شاخصهای کلیدی (PnL، شاپر، روند، هشدارهای ریسک).
* ایجاد گزارشهای دورهای برای تحلیل عملکرد استراتژی و تصمیمگیری برای بازآموزی.

توضیح جامع ساختار استراتژی معاملات فیوچرز مبتنی بر یادگیری ماشینی با بهبود مستمر

استراتژی ما یک سامانه چندبخشی و پیشرفته برای معاملات فیوچرز است که هدف آن بهرهبرداری از مزایای یادگیری ماشینی و یادگیری تقویتی به همراه مدیریت ریسک دینامیک و بهبود مداوم عملکرد میباشد. این ساختار بر مبنای دادههای لحظهای، تحلیل دقیق تغییرات بازار و تصمیمگیری مبتنی بر ترکیب چند مدل طراحی شده است.

1. بخش دادهها و پیشپردازش

* دریافت دادههای بازار به شکل لحظهای (Tick Data) همراه با دادههای دفتر سفارش (Limit Order Book)، اندیکاتورهای تکنیکال، اخبار و دادههای احساسی (Sentiment).
* پیشپردازش شامل نرمالسازی، ساخت فیچرهای تخصصی (مثلاً imbalance در سفارشات، نوسانات محلی، spread و غیره) و ذخیره آنها در یک Feature Store با قابلیت دسترسی کمتاخیر برای مدلها.

2. تشخیص تغییرات و نوسانات بازار

* ماژولی برای تشخیص تغییرات ناگهانی و روندهای نوظهور با الگوریتمهای حساس مانند CUSUM، BOCPD و مدلهای مبتنی بر LSTM یا Transformer برای anomaly detection.
* تفکیک تشخیص سریع (برای واکنشهای عملیاتی و هشدارهای فوری) و تشخیص دقیقتر (برای بازآموزی و تحلیل عمیقتر).

3. مجموعه مدلهای یادگیری ماشینی و یادگیری تقویتی

* مدلهای پیشبینی متنوع: شامل مدلهای زمانسری مثل LSTM و Transformer، مدلهای گراف مانند GNN برای تحلیل ارتباط نمادها، و مدلهای قاعدهای برای پایش ریسک پایه.
* یادگیری تقویتی محافظهکار (Offline RL) با الگوریتمهایی مانند PPO یا SAC برای بهینهسازی مستمر سیاست معاملاتی با کنترل ریسک و جلوگیری از overfitting.
* ترکیب مدلها به صورتی که بتوان سیگنال خروجی را بر اساس وزنهای دینامیک تنظیم شده با توجه به شرایط بازار و دقت مدلها بهینه کرد.

4. لایه ترکیب خروجی (Ensembling & Decision Fusion)

* استفاده از meta-learner برای تلفیق خروجی مدلها با روشهایی مانند stacking، weighted voting، Bayesian Model Averaging و Kalman filter.
* قواعد محافظهکارانه برای جلوگیری از تصمیمات پرریسک در صورت عدم اجماع مدلها (مثل نیاز به اجماع بالای N از K مدل یا محدودیت confidence).

5. اجرای سفارش و مدیریت ریسک

* مدیریت اندازه موقعیت مبتنی بر هدفگذاری نوسان (Volatility Targeting) و قواعد محافظهکارانه مانند stop-loss پویا.
* محدودیت اکسپوژر کل و محدودیتهای اجرایی سفارش بر اساس venue و latency.
* سیستمهای هشدار و kill-switch برای واکنش سریع به شرایط بحرانی و کاهش زیان.

6. چارچوب تست، ارزیابی و شبیهسازی

* event-driven backtesting با شبیهسازی دقیق اجرای سفارش و ارزیابی در شرایط مختلف بازار (روند، رنج، پرنوسان).
* استفاده از تکنیکهای walk-forward evaluation، Monte Carlo simulation، و تستهای استرس (مانند Flash Crash).
* معیارهای مالی و یادگیری ماشینی مانند Sharpe Ratio، Max Drawdown، ROC-AUC، و Precision@k برای پایش مستمر عملکرد.

7. بازآموزی و بهروزرسانی مدلها

* سیستم کنترل کیفیت و trigger-based برای بازآموزی مدلها بر اساس شاخصهایی مانند افت Sharpe یا افزایش خطا.
* استفاده از warm-start و transfer learning در فرآیند retraining برای حفظ پایداری و جلوگیری از فراموشی.
* ترکیب دادههای جدید و قدیمی (replay buffer) برای تضمین یادگیری کافی و متوازن.

8. زیرساخت فنی و ابزارها

* زبان برنامهنویسی Python با استفاده از کتابخانههایی مانند TensorFlow، PyTorch، Stable-Baselines3، scikit-learn.
* زیرساخت داده realtime بر پایه Kafka و Feature Store مثل Feast یا پیادهسازی داخلی.
* ابزارهای backtesting مانند Backtrader و چارچوبهای MLOps جهت اتوماسیون چرخه مدل.
* داشبوردهای مانیتورینگ realtime و سیستمهای هشدار با استفاده از Slack، PagerDuty یا SMS.

این ساختار جامع، انعطافپذیری، ایمنی و دقت لازم را برای یک استراتژی معاملات فیوچرز مبتنی بر هوش مصنوعی فراهم میآورد. با بهرهگیری از چند مدل متنوع و سیستمهای بازخورد و بازآموزی، سیستم قادر به تطبیق با تغییرات بازار و بهبود مداوم خواهد بود.

یک ساختار شماتیک کلی و ساده شده از کد نهایی استراتژی شما که مبتنی بر یادگیری ماشینی برای معاملات فیوچرز است، به صورت ماژولار و قابل توسعه در پایتون به شرح زیر است. این ساختار شامل بخشهای اصلی pipeline داده، محاسبه فیچرها، مدلسازی، ترکیب خروجی و اجرای سفارش به همراه مدیریت ریسک و بازآموزی است:

python

*# فایل: futures\_ml\_strategy.py*

*# بخش 1: دریافت و پیش‌پردازش داده‌ها*

**class** DataPipeline:

**def** \_\_init\_\_(self, data\_source):

self.data\_source = data\_source

self.raw\_buffer = []

**def** ingest\_data(self):

*# اتصال به منبع داده (WebSocket, API, فایل) و ذخیره در بافر*

**pass**

**def** preprocess(self):

*# پاکسازی، نرمال‌سازی، ساخت فیچرهای پایه (SMA, imbalance, VWAP...)*

**pass**

**def** save\_features(self):

*# ذخیره فیچرها در حافظه سریع (Redis) و همچنین ذخیره آفلاین (مثلا Parquet)*

**pass**

*# بخش 2: محاسبه فیچرهای پیچیده، مخصوصاً imbalance وزند‌هی شده*

**def** calculate\_weighted\_imbalance(order\_book):

*# استفاده از داده‌های چند سطح دفتر سفارش، محاسبه imbalance وزند‌هی شده*

**pass**

*# بخش 3: مدل‌های یادگیری ماشینی (Supervised و RL)*

**class** MLModels:

**def** \_\_init\_\_(self):

*# بارگذاری یا تعریف مدل LSTM، Transformer یا GNN*

**pass**

**def** predict(self, features):

*# پیش‌بینی سیگنال بازار*

**pass**

**class** RLAgent:

**def** \_\_init\_\_(self):

*# بارگذاری یا تعریف الگوریتم RL (مثلا PPO)*

**pass**

**def** select\_action(self, state):

*# تصمیم‌گیری بر اساس سیاست RL*

**pass**

**def** update\_policy(self, feedback):

*# بازآموزی مدل بر اساس داده‌های جدید*

**pass**

*# بخش 4: ادغام خروجی مدل‌ها (Ensembling)*

**def** ensemble\_decision(supervised\_output, rl\_output, weights):

*# ترکیب سیگنال‌ها با وزن‌دهی دینامیک و تصمیم‌گیری محافظه‌کارانه*

**pass**

*# بخش 5: اجرای سفارش و مدیریت ریسک*

**class** TradeExecutor:

**def** \_\_init\_\_(self):

*# اتصال به API بروکر، مدیریت اندازه موقعیت، stop-loss و kill-switch*

**pass**

**def** execute(self, decision):

*# ارسال سفارش‌ها و مدیریت پوزیشن‌ها*

**pass**

**def** risk\_management(self):

*# کنترل ریسک و هشدارها*

**pass**

*# بخش 6: حلقه اصلی استراتژی*

**def** main\_loop():

data\_pipeline = DataPipeline(data\_source='exchange\_api')

ml\_models = MLModels()

rl\_agent = RLAgent()

executor = TradeExecutor()

**while** True:

data\_pipeline.ingest\_data()

features = data\_pipeline.preprocess()

data\_pipeline.save\_features()

supervised\_pred = ml\_models.predict(features)

rl\_action = rl\_agent.select\_action(features)

decision = ensemble\_decision(supervised\_pred, rl\_action, weights={'supervised':0.6, 'rl':0.4})

executor.execute(decision)

executor.risk\_management()

*# شرایط و triggerهای بازآموزی*

**if** check\_retraining\_needed():

rl\_agent.update\_policy(feedback\_from\_trades())

*# کمی تأخیر متناسب با فرکانس داده‌ها*

sleep(interval)

*# تابع کمکی تشخیص نیاز به بازآموزی*

**def** check\_retraining\_needed():

*# بررسی معیارهایی مثل افت Sharpe یا افزایش خطا*

**return** False

*# تابع کمکی برای بازخورد از معاملات گذشته جهت آموزش RL*

**def** feedback\_from\_trades():

*# جمع‌آوری داده عملکرد جهت بهبود سیاست RL*

**return** {}

*# نقطه ورود برنامه*

**if** \_\_name\_\_ == "\_\_main\_\_":

main\_loop()

**توصیف کلی ساختار:**

* **DataPipeline** مسئول دریافت دادههای تیک و دفتر سفارش به صورت لحظهای، پیشپردازش و استخراج فیچرهای پایه مثل SMA، imbalance و VWAP و ذخیره آنها برای استفاده مدلها است.
* توابع مانند **calculate\_weighted\_imbalance** میتوانند در این pipeline برای محاسبه فیچرهای پیچیدهتر استفاده شوند.
* کلاسهای **MLModels** و **RLAgent** نقش مدلهای پیشبینی و یادگیری تقویتی را اجرا میکنند که خروجی آنها به روش ensemble برای تصمیمگیری ترکیب میشود.
* **TradeExecutor** مدیریت ارسال سفارش، کنترل ریسک، پیادهسازی حد ضرر و هشدارهای ضروری را برعهده دارد.
* حلقه اصلی برنامه به شکل پیوسته داده را دریافت، پردازش، مدلسازی، اتخاذ تصمیم و اجرای سفارش را انجام میدهد و در صورت نیاز مدلها را بازآموزی میکند.

این ساختار ماژولار و سادهشده است ولی بر پایه توصیهها و معماریهای روز سال ۲۰۲۵ توسعه یافته و میتواند به راحتی گسترش و شخصیسازی شود. اگر بخواهید میتوانم برای هر بخش نمونه کد دقیقتر و کاربردیتر بر اساس کتابخانههای معرفیشده (مثل pandas, polars, Stable-Baselines3, FastAPI و Redis) آماده کنم.