1. Veri Kaynakları ve Özellik Mühendisliği

1.1. Mevcut alternative_data.py'yi Zenginleştirme
Şu anki hali:
Google Trends (EURUSD, USDJPY, GBPUSD → her 15 dakikada güncelleniyor)
Twitter Sentiment ("forex" hashtag'i üzerinden global İngilizce tweet'lerin compound skoru)
Geliştirme önerileri:
Çok dilli ve çok platformlu duygu analizi:
Sadece "forex" etiketi yerine "#forextrading", "#EURUSD" gibi spesifik etiketleri de takip et. Türkçe, İspanyolca, Arapça gibi farklı dillerdeki tweet'lere de bak. (Tweepy sorgularını lang:tr, lang:es, lang:ar şeklinde genişletebilirsin.)
Reddit, StockTwits, Investing.com gibi platformlardan da veri al. Örneğin Reddit'in r/forex ya da r/algotrading toplulukları, "pushshift.io" API'si üzerinden toplanabilir. Bu sayede tek platforma bağlı kalmamış olursun.
Konu Tabanlı Duygu Filtrelemesi (Topic Modeling):
Vader yerine, daha ince ayrıştırma yapan bir yöntem kullan. Örneğin Türkçe Reddit kirpiklerinden gelen veriye LSTM tabanlı bir "duygu sınıflandırıcı" eğiterek, "boğa haberleri mi, ayı mı" kategorisini net çıkar.
Geliştirdiğin model: TextCNN ya da DistilBERT (küçük bir transformer) olabilir. Training datası için geçmiş tweet/değerlendirmeleri elle etiketleyebilirsin. Böylece sadece compound skor değil, "boğa (bullish)", "ayı (bearish)", "nötr (neutral)" gibi üçlü bir çıktı alırsın.
Haber Kaynağı Çeşitliliği:
Şu an sadece NewsAPI üzerinden alıyorsun. Bu API'nin sınırlamaları (küçük haber başlıkları, gecikmeler) var. Bunun yanı sıra:
ForexFactory ve Investing.com Economic Calendar (REST / kebab) entegrasyonu ekle. Yüksek etkili haberleri (ör. NFP, Fed Kararı, CPI) "yakala".
"Panic keywords" listesini genişlet: Sadece "crash, dump, volatility" değil, "Fed açıklaması, faiz artırımı, varlık alımı, varlık satımı" gibi terimleri de ara.
Haber başlıklarına ek olarak RSS beslemeler (Bloomberg, Reuters) parçalayabilirsin.
On-Chain ve Akıllı Para İşaretçileri (Smart Money Signals):
Özellikle kripto paralar yerine forex odaklısın ama "kur paritelerindeki büyük banka pozisyonları"nı gösteren ücretsiz/ücretli API'ler var. Örneğin CFTC'ın haftalık "Commitment of Traders" raporunu (COT data) çek.
OTC bankalar veya hedge fonlarının "net long/net short" pozisyonlarını Google Sheets'ten çekip botun kararlarına ekle.
Derinlik Verisi (Order Book / DOM):
Mümkünse MT5'ten "Market Depth" (Level II) verisi al. Bir paritede anormal biriken emirler (büyük lot ask/bid duvarları) varsa, bu "manipülasyon" veya "yaklaşan volatilite" sinyali olabilir.
Eğer VPS/MT5 ile bağlantın bu veriyi veriyorsa, alt bantta bu order book verisini de normalize ederek ("bid ask ratio" gibi) state vektörüne ekleyebilirsin.
1.2. Çok Zamanlı ve Çok Ölçütlü Özellikler

Şu anki hali:
M1 (1 dakikalık), H1 (1 saatlik) ve H4 (4 saatlik) veri çekiliyor. LSTM H1 bazlı "bullish/bearish" rejim tespiti.
Geliştirme önerileri:
Daha İnce Zaman Aralıkları (Tick Data / 5S / 15S):
FTMO'da ultrakısa zaman dilimlerinde (örneğin 5 saniye) tarama yapmak, scalping fırsatlarını yakalamak için önemli. Mümkünse "MT5.copy_ticks_from" fonksiyonunu kullanarak son X saniyenin tick'lerini topla. Onlardan "son 5S'deki volatitiy spike" gibi ek filtreler oluştur.
Multi-Timeframe Özellik Birleştirme (MTF data fusion):
Sadece H1'den LSTM rejim tespiti değil, H4'ten trend açısı (örneğin H4'deki RSI, H4 MACD divergence) çıkar.
M1'de karar verirken:
M1'deki kısa vadeli momentum (RSI(7), CCI(14), Momentum(5)).
H1'deki orta vadeli trend (H1 RSI(14), H1 MACD histogram).
H4'teki uzun vadeli trend (örneğin H4 EMA(50) vs EMA(200) kesişmesi).
Bu üçünü normalize ederek (0–1 aralığına) state'e ekle. Böylece DQN, tek timeframe yerine "temel çerçeveyi" de bilir.
Teknik / Temel Kombinasyonu:
Paritenin işlem gördüğü ülke ekonomileriyle (örneğin EURUSD için Euro Bölgesi ve ABD CPI, İsizlik, PMI) arasında regresyon analizi yap. Eğer fundamental veriler "uyumsuz" (yani fiyat hareketi sınırlı veya ters) ise DQN aşırı açılmaya teşebbüs etmesin.
Bunun için "pandas_datareader" yerine "yfinance" veya "fredapi" kullanabilirsin.
2. Model İyileştirmeleri ve Öğrenme Algoritmaları

2.1. DQN'den Rainbow'a: Deneyim Yeniden Oynama ve Gelişmiş RL
Şu anki hali:
Double DQN + Dueling yapısı var. Replay buffer (öncelikli değil), ε-greedy keşif.
Geliştirme önerileri:
Öncelikli Deneyim Yeniden Oynama (Prioritized Experience Replay):
ReplayBuffer'ı sadece FIFO queue olarak kullanmak yerine, önemli anları (yüksek TD hatası) daha sık örnekle. Böylece öğrenme hızlanır.
[Schaul vd., 2015] makalesindeki "Proportional Prioritization" algoritmasını Python tarafında entegre edebilirsin.
Distribütüyonel RL (Categorical DQN veya QR-DQN):
Q değerlerini tek bir sayı yerine bir dağılım olarak modellemek, risk duyarlılığını arttırır. "Mükemmel bir kazanç" yerine "en yüksek %5'lik kayıp riski"ni minimize edebilirsin.
TensorFlow'da bir "categorical DQN" mimarisini (51 atomlu C51) kurarak, DQN'in yerine geçirebilirsin.
Multi-step Returns (n-step target):
Şu anki DQN, bir adım sonrası getiriye bakıyor. 3–5 adımlık Gt = r_t + γr_{t+1} + ... şeklinde daha uzun horizon hedefler, sinyal ağacını stabil hale getirir.
ReplayBuffer içine "state, action, reward, next_state, next_next_state, ..." olarak ekleyip, n_step ödüller hesaplanabilir.
Noisy Nets ve Parameter Noise:

Epsilon-greedy yerine, ağ yapısına “Noisy Networks” katmanı ekleyerek (örneğin tfp.layers.NoisyDense veya kendi “NoisyLinear” implementasyonu) keşfi parametrik hale getir.
Bu sayede, her epizotta epsilon ayarıyla uğraşmak yerine, ağa “hangi ağırlıklar rastgeleleştirildi” diyebilirsin.
Rainbow DQN = Dueling + Double + Prioritized + Multi-step + Noisy + Distribütüyonel:
Tüm bu bileşenleri tek bir “RainbowAgent” sınıfı altında toplamak, teoride “en iyi DQN” performansını getirir. Kütüphanelerde (örneğin “tensorflow_agents” veya “stable_baselines3”) örnek altyapılar var; kendi özel ayarlarına uyarlayarak daha hızlı devreye alabilirsin.
Alternatif: Proximal Policy Optimization (PPO) veya A2C/A3C:
DQN tabanlı değil, politika tabanlı (actor-critic) yaklaşımlara geçebilirsin. PPO, işlem ortamının sürekli değişken dinamiklerine daha hızlı adapte olabilir.
Özellikle “continuous action space” yerine “al/boş/sat” üçlü discrete action olsa da yine de PPO’yu uygulamak mümkün. “Stable Baselines3” içinden PPO’yu çağırıp, env alt yüzeyini hazırla.
Örneğin her M1 mum kapanışında bir adım at, ödül olarak “profit - risk_penalty” ver. PPO ile daha keskin (stabil) öğrenme elde edebilirsin.
2.2. Rejim Tespiti: LSTM’den Üst Seviyeye
Şu anki hali:
H1 kapanış verileriyle 16 nöronlu basit bir LSTM. “Return > 0” olarak etiketleme.
Geliştirme önerileri:
Daha Geniş Girdi Dizisi ve Çoklu Özellik:
Sadece “close” yerine H1’de ['open', 'high', 'low', 'close', 'volume'] kullan. Bunları MinMaxScaler ile normalize et. “Close Returns” modeline ek olarak “High-Low Range” (volatilite bilgisi), “RSI” vb. ekle. Bu şekilde LSTM, sadece fiyat yönünü değil, momentum ve volatilite değişkenlerini de öğrenir.
Transformer Tabanlı Zaman Serisi Modeli:
LSTM’nin yerine “Informers” veya “Time Series Transformer” (örn. “TFT – Temporal Fusion Transformer”) kullan. Bu modeller, uzun vadeli bağımlılıkları daha hızlı kavrar. H1’deki 5000 saat yerine 20000 saatlik veriyi Transformer’a yedir, token-length 100–200 aralığında kaydırmalı pencere (sliding window) kullan.
Regresyon/Eğilim (Trend) Sınıflandırmasından Daha Fazlası:
“Bullish/Bearish” yerine, üç sınıf kullan: “Güçlü boğa (strong bullish) / zayıf boğa / nötr / zayıf ayı / güçlü ayı”. Bu sayede daha incelikli karar mekanizmaları oluşturabilirsin.
Online Learning ve Periyodik Yeniden Eğitim:
Eğitilmiş H1 modeli her hafta/ay otomatik olarak yeniden eğitilsin (cron job veya bot başlatılırken).
Eski ağırlıkları “fine-tune” ederek yeni güncel veriye adapte ettir (“warm start”).
Sinyal Çatışması Açıklığı için Güven Skoru (Confidence Score):
LSTM/Transformer çıktısının sigmoid veya softmax skorunu al. Eğer “regime_confidence < 0.6” ise “nötr” de; aksi durumda tam “bullish/berish” olarak tespit et. Bu güven skoru, DQN’e state olarak eklenebilir.
2.3. Enchilada: Ensemble (Topluluk) Modelleri

Neden?
Bir tek modelin yanılma riski vardır. Ensemble ile hata olasılığını azaltır, daha genelleyici sonuç elde edersin.
Nasıl?
Farklı RL Algoritmalarının Odağı:
Bir tarafta Rainbow DQN, diğer tarafta PPO, bir diğer tarafta A2C. Her biri ayrı agent olarak parallel eğitilsin (aynı offline veride backtest).
Eğitim sonunda "meta-learner" (örneğin küçük bir MLP) hangi agent'ın o anki state'de (özellik vektörüne bakarak) daha güvenilir olduğunu tahmin etsin.
Sonra o agent'ın eylemi (buy/hold/sell) dominansin olsun.
Majority Voting / Weighted Voting:
Her agent'ın Q veya policy çıktısı bir skor olarak alınıp, en yüksek ortalama skor kararına göre trade açılsın.
Uzun vadede hangi agent hata yapıyorsa, performansına göre ağırlığı azaltılsın.
3. Risk ve Portföy Yönetimi

3.1. FTMO Kurallarını Kesin Sağlama
Şu anki hali:
Günlük %5, toplam %10 limit var, aynı anda 1 pozisyon (hedge yasak).
Geliştirme önerileri:
Dinamik Maksimum Pozisyon Sayısı:
FTMO'da "aynı anda en fazla 1 pozisyon" kuralı var ama "farklı semboller" için bazen loophole bulunabiliyor. Botu, "hesap sabit + floating PnL" takibini yapacak şekilde güncelle. Yani canlıda mt5.account_info().equity + mt5.positions_total() verilerini okuyan periyodik fonksiyon getir.
Gerçek Zamanlı Equity/Drawdown Takibi:
Şu an risk manager'e sadece backtest sonuçlarıyla geliyor bilgi. Canlıda, mt5.account_info().balance, mt5.account_info().equity'den anlık çekerek, "floating drawdown" (% olarak) kontrol et. Eğer yıllık "max daily drawdown" %5'i uzaktan aşacak gibi görünüyorsa (örneğin floating zarar > x), pozisyonları otomatik kapat.
VaR (Value at Risk) Hesaplama:
Her gün M1 kapanışında, portföyün "1 günlük %95 güven aralığındaki potansiyel en kötü kaybını" (Parametrik VaR veya Monte Carlo) hesapla. Eğer VaR %2.5'i aşıyorsa, o gün yeni pozisyon açma.
Basitçe portfolio_return_mean ve portfolio_return_std bulup VaR = mean + 1.65*std formülüyle bir eşik koy.
Pozisyon Büyüklüğünü Tasfiye Olasılığına (Probability of Ruin) Göre Ayarla:
Basit risk yönetimi: her işlem maksimum "hesap büyüklüğünün %1'i" kadar riske izin verse de, PoR (Probability of Ruin) teorisine göre puanı hesapla. Eğer artan volatilite, ardışık küçük kayıplar PoR'u "≥%5" eşiğine çıkarırsa, lotu aşağı çek veya gün ortasında otomatik "hedge" yap.
Paralel Portföy Yönetimi (Asset Allocation):
Şu an üç ayrı sembol, ayrı ayrı risk hesaplaması var. Bunun yerine "global stop loss" da tanımla:
Hesap büyüklüğünün %3'ü kadar toplam floating zarar varsa, tüm pozisyonları kapat.

Her sembol için risk katsayısını dinamik ayarla: Örneğin EURUSD'e %40, USDJPY'ye %30, GBPUSD'ye %30 tahsis et. Böylece "bir sembolde zarar diğerini etkilemesin" diye.
Otomatik Sosyal Lokavt (Social Lockout):
Demo/gerçek geçişinde, FTMO mücadelesi sırasında, insan beklenen bir volatilite hatasında müdahale edebilsin. Bot, kritik saatlerde (örneğin Fed toplantısı anında, BBC'de flaş haber çıktığında) hiçbir pozisyon açmasın (Zaman Çerçevesi: haber saatinin ±15 dakika). Bunu "economic calendar" entegrasyonu ile yapabilirsin.
3.2. Dinamik Lot ve Risk Katmanlama
Şu anki hali:
ATR'e göre lot mult. (yüksek vol → lot×0.5, düşük vol → lot×1.5)
Geliştirme önerileri:
Kelly Kriteri Temelli Lot Hesabı:
"Her işlemde beklenen başarı oranını" LSTM rejim modeli + backtest sonuçlarına göre hesapla. Diyelim "şu anda bullish rejim, başarısı %60"; Kelly formülünü uygula:
$f^* = W - \frac{R}{2}$
(W = speech of winning, R = risk/reward)
$f^* = \frac{W}{R}$

(W = speech of winning, R = risk/reward)
Örneğin kazanma olasılığı %60, risk/ödül 1:2 ise,
$f^* = 0.6 - \frac{0.4}{2} = 0.4$
$f^* = 0.6 - 0.4/2 = 0.4$. Bu da "hesap sermayesinin %40'ı" anlamına gelmez, çünkü forex pip olarak hesaplanır. Ama "lot size = account_balance × f^* / fiyat" mantığıyla dinamik ayar yap.
Pozisyon Merdivenleme (Scaling In/Out):

İlk sinyalde yarım lot, fiyat senin lehine hareket etmeye devam ederse kalan yarım lot ile tekrar girecek şekilde bir “ladder entry” yap.
TP’ye yaklaşınca kademeli kâr al (örneğin %50 lot kapanınca, kalan %50 için trailing stop geç).
Kar Kilitleme (Profit Lock-in):
Pozisyon açıldıktan sonra +X pip kar elde edildiyse (örneğin +10 pip), otomatik olarak SL’i entry price’a çek. Yani kayıp riskini sıfırla. Sonra +20 pip olunca SL’i “+5 pip” e taşı. Bu mantığı “sl_multiplier” ve “tp_multiplier” ile güncelleyebilirsin.
Kaybı Hızla Azaltma (Loss Guard Geliştirmeleri):
3 zararlı işlem → 1 saat “sleep” yerine, “zamanla azalan bekleme” (adaptive dormancy): Eğer art arda zararlı işlemler artıyorsa, bir sonraki işlem için bekleme → 30 dk, 15 dk, 5 dk şeklinde düşer.
Eğer “toplam floating drawdown > hesap bakiyesinin %2’si” ise, otomatik “hedge” sinyali çıkar. Yani “aynı paritede ters pozisyon aç” (kısmen veya tam).
Çok Katmanlı Risk Puanı (Composite Risk Score):
ATR spike, yüksek korelasyon, LSTM güven skoru, sosyal medya paniği, haber volatilite endeksi gibi unsurları 0–1 aralığında topladığın alt risk skorlarıyla “composite risk score” oluştur.
Eğer bu skor “>0.7” ise, pozisyon açma. (Benim anlamımda 0 = risksiz, 1 = aşırı riskli.)
4. Canlı İşlem Altyapısı ve Mimarisi

4.1. Gerçek Zamanlı Veri Hattı (Data Pipeline)
Şu anki hali:
“run_backtest” sırasında paralel olarak M1/H1/H4 verileri alınıyor.
Canlıda her dakika “copy_rates_from_pos” çağrısı yapılıyor.
Geliştirme önerileri:
Webhook + Event-Driven Veri Alımı:
Sadece “her 60 saniyede bir” diye poll etmek yerine, MT5’nin “OnTick” olaylarına abone ol. Python tarafında mt5.copy_ticks_from ile tetiklenen bir “callback” fonksiyonu yaz. Bu sayede “anlık olarak tick verisini” işleyip, M1 mum kapanışını biraz daha gerçek zamanlı alırsın (60s yerine ~1–2s gecikmeyle).
Kafka / Redis Yayın (Pub/Sub) Katmanı:
Botunu ölçeklendirip, farklı modüllerin (veri toplama, sinyal üretme, risk yönetimi, order execution) birbirinden bağımsız çalışmasını istiyorsan, “message broker” kullan. Örneğin:
Veri Toplama Servisi → M1 tick’i geldiğinde “kuyruğa” push.
Özellik Hesaplama Servisi → Kuyruktan al → teknik göstergeleri hesapla → “state queue”a gönder.
Agent (RL) Servisi → “state queue”dan al → action belirle → “order queue”a gönder.
Order Execution Servisi → “order queue”dan al → MT5 order_send ile işlemi gerçekleştir.
Risk Manager Servisi → Her kapanan pozisyonda tetiklenir, drawdown, VaR hesaplar.
Böylece bir servis çökse bile diğerleri çalışmaya devam eder. Ayrıca ileride

başka agent’lar eklemek isteyen birinin altyapıyı yeniden kurgulaması kolaylaşır.
Latensi Minimize Etmek için VPS:
FTMO challenge’da “0.01 saniye gecikme” bile kritik olabilir.
Botu mutlaka İstanbul veya Frankfurt lokasyonlu bir VPS’e taşı. MT5 Cloud VPN entegrasyonu kullanarak “en düşük ping” yoluyla broker sunucusuna bağlan.
Yüksek Erişilebilirlik (High Availability):
Botun ana süreci çöktüğünde hemen yedek bir instancia devreye girsin. Basitçe “systemd” ile Restart=on-failure ayarlayabilirsin.
Bir de “ana logdosyası”nı (ftmo_trading_bot.log) haftalık olarak rotasyonlu (logrotate) tut. Böylece geçmişe dönük hata incelemesi kolaylaşsın.
4.2. İnsan-Makine Arayüzü (Human-in-the-Loop)
Neden?
“Tamamen otomatik” bir sistem her zaman yanılmaz demek değildir. Özellikle beklenmedik piyasa çöküşleri, kara swan olayları, veri kesintileri vs. için insan onayı gerekebilir.
Geliştirme önerileri:
Web Tabanlı Dashboard:
Grafana + Prometheus:
Bot metriklerini (loss, reward, epsilon, günlük PnL, güncel drawdown, open position sayısı) Prometheus’a push et. Grafana’da grafikleri canlı gözetle. İnsan piyasanın garip gittiğini görürse “manual override” yapabilir.
Flask/Django + Plotly Dash:
Eğer daha özelleştirilmiş bir şey istersen, Python tabanlı bir sonraki adım. Flask arkasında bir web sunucu, Plotly Dash ile:
Canlı eşik grafikleri: Günlük kâr/zarar eğrisi.
Open Position Listesi: Hangi sembolde, hangi lot, SL/TP, açılış fiyatı.
Trade History: Son 50 trade’in RSI, MACD, girdiği fiyat, çıktığı fiyat, GHN skorları vs.
“Manual Override” butonu: “Şu anda bot «GBPUSD» için alım yapmasın” diyebilirsin. Bu komut bir JSON içinde custom_params['disable_GBPUSD'] = True şeklinde ana programa geçer.
Telegram Arayüzü (Command & Control):
Şu an sadece send_telegram_message var ama “/pause”, “/resume”, “/stats” komutlarını işleyen bir Telegram bot ekle.
/stats → “Bugün 5 işlem yapıldı, +120 USD kâr. En son işlem EURUSD, +10 pip.”
/pause → Botu geçici olarak durdur.
/resume → Botu kaldığı yerden devam ettir.
/override EURUSD sell → Bot şu anda EURUSD’de satma izni versin.
Bunu yapmak için “telegram.ext” içinden CommandHandler ve Updater kullanabilirsin.
E-mail ve SMS Uyarıları (Fallback Kanal):
Telegram’a erişim kesilirse, e-posta veya SMS (Twilio) üzerinden kritik uyarılar gönder:
“Günlük zarar limiti aşıldı”
“MT5 bağlantısı kayboldu”
“Yüzde 2 floating drawdown tetikledi”

Böylece her zaman erişim kanalı olur.
5. Test, Optimizasyon ve Dağıtım Süreçleri

5.1. Overfitting'i Engellemek için İleri Düzey Test Metodları
Şu anki hali:
Training: 2015–2023, Testing: 2024.
Geliştirme önerileri:
Walk-Forward Analizi (WFA):
Örneğin:
2015–2017 ile eğit → 2018 test
2015–2018 ile eğit → 2019 test
2015–2019 ile eğit → 2020 test
... → 2024 test
Her "eğitim–test" aşamasında en iyi hiperparametreleri seç, sonra ileriki döneme geç. Bu sayede parametreler geçmişe "aşırı uyumlu" (overfit) olmaz.
K-Fold Zaman Serisi Çapraz Doğrulama:
Normal k-fold değil, "zaman pencereli" cross-validation: her fold'da test aralığını eğitim aralığından sonra konumlandır. Örneğin 5-Year Train, 1-Year Test olarak slayt.
Out-of-Sample (OOS) ve Out-of-Time (OOT) Dönemler:
Backtest'te "en yeni veriler"i asla eğitimde kullanma (örneğin 2024 yılının son çeyreği). Sadece final demo/verification için "live-replay" modunda o dönemi kullan.
Monte Carlo Simülasyonları & Sensitivite Analizi:
Son 1000 trade'in "şu pip dağılımına göre kümülatif PnL'si" adım adım varyasyonlu simülasyon yap. Hangi pip kayıplarına, hangi korelasyon değişimlerine bot nasıl tepki veriyor?
Python'da numpy.random.multivariate_normal ile pip hareketlerini varyasyonlu çek, backtest loop'unu 1000 kez çalıştır. Her seferinde drawdown, Sharpe, Sortino gibi metrikleri kaydet. Metriklerin varyansını incelersen, gerçek hayatta "bot ne kadar istikrarlı?" sorusunun yanıtını alırsın.
Takım Arkadaşlarıyla Kod İncelemesi (Code Review):
Bot mimarisini JTTWS projesi ekibinden en az bir meslektaşına gözden geçirt. Kodun "temiz kod" (clean code) standartlarına uygun olduğuna emin ol. Özellikle multithreaded/paralel yapıda "race condition" olmayacak şekilde dikkat et.
5.2. Hiperparametre Optimizasyonu
Şu anki hali:
custom_params içinde birkaç parametre el ile verilebiliyor, Genetik koda hazır bir "iskelet" var.
Geliştirme önerileri:
Optuna / Hyperopt Kütüphanesi Kullanarak Boost
Genetik yerine Optuna'nın TPE (Tree-structured Parzen Estimators) algoritmasını kullan.
objective(trial) fonksiyonunda:
rsi_low = trial.suggest_int('rsi_low', 10, 50)
rsi_high = trial.suggest_int('rsi_high', 50, 90)

atr_sl_mult = trial.suggest_float('atr_sl_mult', 1.0, 3.0)
atr_tp_mult = trial.suggest_float('atr_tp_mult', 2.0, 6.0)
learning_rate = trial.suggest_loguniform('learning_rate', 1e-4, 1e-2)
epsilon_decay = trial.suggest_uniform('epsilon_decay', 0.95, 0.999)
# vs...
return backtest_sharpe_ratio(...)
50–100 deneme sonunda en iyi parametreleri best_params = study.best_params ile alıp, final testte kullan.
Multi-Objective Optimizasyon (Sharpe + Max Drawdown):
Sadece kâr değil, Sharpe'ı (veya Sortino) optimize et. Aynı anda "maksimum drawdown"'ı minimum tut. Optuna, study = optuna.create_study(directions=["maximize", "minimize"]) ile çok amaçlı arama yapabilir.
Bayesyen Optimizasyon (Gaussian Process):
Eğer Optuna istemezsen, skopt (scikit-optimize) kullanarak GP tabanlı arama yap. Özellikle parametrelerin sürekli ve dar bir aralıkta olduğu durumlarda GP daha hızlı yakınsama sağlar.
5.3. Sürekli Entegrasyon ve Dağıtım (CI/CD)
Neden?
Kod güncellemeleri sırasında "bir özellik bozuldu, canlıda takıldı" gibi riskleri en aza indirmek için.
Geliştirme önerileri:
GitHub Actions ile Otomatik Testler:
Kod değiştiğinde:
Unit test: "calculate_indicators"'ın doğru çıktısı mı? "RiskManager" limitleri aşıyor mu? "generate_signal" edge-case'lerde None mu döndürüyor?
Linting/Formatting: flake8 ve black ile kod kalitesi.
Başarılı olduğunda main branch'e merge et.
Dockerize Etme:
Python versiyonu, bağımlılıkları (ta-lib, TensorFlow, vs) her makinede aynı olsun.
Dockerfile örneği:
```
FROM python:3.9-slim
RUN apt-get update && \
    apt-get install -y libta-lib0 libta-lib0-dev build-essential
WORKDIR /app
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
COPY . .
CMD ["python", "ftmo_trading_bot.py"]
```
Böylece VPS'te docker-compose ile "indir → ayağa kaldır" basitliği sağlanır.
Versiyon Takibi + Model Kayıt (Model Registry):
Eğitilen her DQN/PPO modeli ve LSTM/Transformer rejim modeli "models/" klasörüne "datetime_model_adı.h5" gibi versiyonlu kaydedilsin.
Hangi versiyonun FTMO sınavında en iyi performans verdiğini geri izleyebilmek için bir küçük models/registry.json dosyası tut:
```
{
  "2025-06-01_18:00_ftmo_dqn": {
```

```
    "sharp_ratio": 1.25,
    "max_drawdown": 3.8,
    "notes": "En iyi test sonucu."
  }
}
```

6. İnsan ve Yapay Zekâ İşbirliği: Son Aşama

6.1. İnsan Karar Düğmeleri (Human Override)

Neden?

Her algoritma “büyük siyah kuğu” (black swan) olayını öngöremeyebilir. İnsan uzmanlığının kritik noktada devreye girmesi, zararları asgariye indirir.

Geliştirme önerileri:

“Manual Pause” ve “Manual Resume” Butonları:

Web arayüzde gerçek zamanlı “Pause” / “Resume” görebileceğin bir switch ekle. Buna basıldığında ana döngü bir “global flag” (örneğin is_paused=True) ile kontrol edilerek hiç işlem açılmasın.

Önemli Haber Anında Bildirim + Onay:

Örneğin Fed kararının yayınlanmasına 10 dakika kala Telegram’dan “Fed toplantısı 10 dk sonra. Bot pozisyon kapatsın mı? (E/H)” sorusu gönder. İnsan “E” derse tüm pozisyonları kapatsın.

Bunu yapmak için, NewsAPI / ForexFactory takvimi parse eden ek bir coroutine yaz:

```
async def economic_event_watcher():
    while True:
        now = datetime.utcnow()
        events = newsapi.get_everything(q="FOMC meeting",
from_param=now.strftime("%Y-%m-%dT%H:%M:%SZ"), to=...)
        # vs. ya da Investing.com calendar parse
        if büyük_etkinlik_varsa:
            asyncio.run(send_telegram_message("Fed toplantısı 10 dk sonra.
Onaylıyor musunuz?"))
            # Bekle 5 dakika insan cevabını
        await asyncio.sleep(300)
```

6.2. Sürekli Öğrenme ve İnsan Geri Bildirimi (Reinforcement Learning + Human Feedback, RLHF)

Neden?

Bot canlıda işlem yaptıkça “insan herhangi bir yerde hata gördü mü?” sorusunu sormadan öğrenmeye devam edemez. İnsan yorum veya “Bu sinyal hatalıydı” gibi doğrudan geri bildirimler, modelin self-correction (öz-düzeltme) yeteneğini artırır.

Geliştirme önerileri:

“Feedback Loop” Arayüzü:

Web dashboard’da her kapanan işlemden sonra “Bu karar doğru muydu? (👍 / 👎)” sorusu sorulsun.

İnsan “👎” yaparsa, o trade “negative_example” olarak kaydolup, bir “HumanCritic” sınıfına eklenir. Belki “örneğin, RSI çok düşük, news negatifken hala almışsın” gibi kısa not alan bir alan da ekleyebilirsin.

RLHF (Reinforcement Learning from Human Feedback):
Her gün toplanan “olumlu” ve “olumsuz” örnekleri bir sınıflandırıcı (örn. küçük bir MLP) eğitir: “Yanlış karar” / “Doğru karar”.
Bir sonraki LSTM/Rejim modeline bu çıktıyı bir “advisor feature” olarak ekle: “İnsan bu durumu %80 hata olarak işaretledi”.
DQN’de, eğer “human_feedback < 0.5” (yani insan beğenmedi) ise, reward fonksiyonunu -X yap (ceza). Bu sayede RL ajanı, insanların hoşuna gitmeyen kararları daha az verecek şekilde kendini optimize eder.
Yarı-Automatik (Semi-Auto) Mod:
Bot “Öneri” modunda çalışsın: Her sinyalde Telegram’a “EURUSD için AL sinyali. Onaylıyor musunuz? (E/H)” mesajı gitsin. İnsan onaya basarsa bot (“al” komutuyla) açsın. Bu şekilde insan, hem sinyal mantığını izler hem de canlıda müdahale eder.
Özellikle ampirik olarak “robot sinyali X sinyalini gönderdi; insan onayladı ve profit +5 pip” gibi günlük raporlar toplanır. Böylece “insan neden onayladı?” sorusunun cevabı veritabanına kaydedilmiş olur.
7. Özet ve Bir Sonraki Adımlar

Veri ve Özellik Setini Zenginleştir
Multiplatform duygu analizi, RSS beslemeleri, order-book, COT data, ek time frame’ler.
RL Algoritmasını Rainbow & Ensemble’a Yükselt
Prioritized Replay, Distribütüyonel RL, Noisy Nets, multi-agent ensemble.
Regim Tespiti (LSTM → Transformer)
Multi-feature girdi, daha fazla sınıf, ilave doğrulama, online fine-tune.
Risk Yönetimi ve Dinamik Lot
VaR, Kelly, çok katmanlı risk skoru, adaptif dormancy, otomatik hedge.
Canlı İşlem Altyapısını Geliştir (Event-Driven, Kafka, VPS)
“Websocket-like” on-demand veri alımı → latency düşürü, HA yapısına geç.
İnsan-Makine Arayüzünü Tesisi Et (Dashboard, Telegram Commands, Webhook)
Güven artar, kritik anlarda insan durdurabilir/devam ettirebilir.
Test ve Optimizasyon Katmanları Kuralım (Walk-Forward, Cross-Validation, Monte Carlo)
Overfitting’i en aza indir, hiperparametreleri Optuna aracılığıyla tara.
Sürekli Entegrasyon & Dağıtım (CI/CD, Docker, GitHub Actions)
Kod kalitesi, otomatik testler, versiyon takibi, reproducibility.
İnsan Geri Bildirimi Döngüsünü (RLHF) Ekleyerek Self-Correction Sağla
Web arayüzde anlık insan dönüşünü topla, modelin ödül fonksiyonunu güncelle.
Bu adımları hayata geçirmek, elindeki botu hem teknik hem de stratejik açıdan “insan kalibrasyonu + yapay zekâ optimizasyonu” ekseninde maksimum noktaya taşıyacak.

Uygulamaya Başlarken Önerilen Öncelik Sırası
1–2 Hafta İçinde:
alternative_data.py’ye Türkçe tweet, Reddit entegrasyonu ekle.
DQN sınıfına “Prioritized Experience Replay” kat.
LSTM rejim modelini birkaç ek özelliğe göre yeniden eğit.

RiskManager'a "VaR hesaplama" modülünü tak.
2–4 Hafta İçinde:
Rainbow DQN'ı vanilla DQN'in yerine geçir.
Optuna entegrasyonunu yap, geri test için hiperparametre arayışına başla.
Dashboard'u basitçe "Flask + Plotly Dash" ile ayağa kaldır; canlı veri gelmesini sağla.
1–2 Ay İçinde:
Transformer tabanlı rejim modelini hayata geçir, önce offline test et.
Ensemble ajan yapısını kur, meta-learner'ı basit bir MLP ile implement et.
Backtest'te Walk-Forward analizini devreye al, sonuçları raporla.
3–6 Ay İçinde:
Webhook + Kafka altyapısını kur, modüler mikroservis mimarisine geçişi tamamla.
RLHF modülünü entegre et; insan geri bildirimini hem sinyal kalitesini hem de risk yönetimini optimize edecek şekilde kullan.
Tam Dockerize edilmiş, GitHub Actions CI/CD pipeline'lı, production-ready bir yapı kur.
Sonuç:
Bu planı adım adım uyguladığında, elindeki DQN+LSTM tabanlı FTMO botu;

Saniyelik tick data ve çok kanallı (multiplatform) haber/duygu entegrasyonu,
Rainbow/Ensemble RL ile artırılmış öğrenme hızı ve kararlılık,
Gelişmiş risk yönetimi (VaR, Kelly, composite risk),
Otomatik "event-driven" altyapı (low-latency) ve
İnsan geri bildirimiyle sürekli adaptasyon (RLHF)
özelliklerine kavuşacak.