Михаил Шардин
личный блог

11 марта 2025, 04:45

Тестирование торговой стратегии с использованием нового индикатора Джона Ф. Элерса на Python для дневных данных Московской биржи

Торговля акциями требует гибкости, особенно когда речь идет о тестировании стратегий технического анализа на прошлых данных. Я выбрал Python и библиотеки backtesting.py и aiomoex, потому что они позволяют анализировать рынок без сложных платформ и ограничений. Python дает свободу автоматизации, backtesting.py обеспечивает удобный и быстрый механизм тестирования стратегий, а aiomoex позволяет скачивать данные напрямую с Московской биржи без привязки к брокеру.

Важно, что backtesting.py получил обновление после четырех лет без обновлений, что делает его актуальным инструментом. И в отличие от MetaTrader, StockSharp, TSLab и Quik, которые работают с Московской биржей, но требуют Windows, если брокер имеет API, то можно запускать скрипт на любом сервере, включая облачные решения и Raspberry Pi.

В этой статье я протестирую самую свежую стратегию теханализа Джона Ф. Элерса (John Ehlers), направленную на устранение запаздывания скользящей средней. Разберемся, как её адаптировать к акциям Московской биржи и протестировать с помощью Python.

Новый индикатор Джона Элерса «устранение запаздывания скользящей средней»

Одна из главных проблем стандартных скользящих средних (SMA) — это запаздывание. Поскольку SMA рассчитывается как среднее за определенный период, её значение всегда отстает от реальной цены, что мешает своевременному входу в сделку.

Джон Элерс предложил решение — прогнозируемая скользящая средняя (PMA, Projected Moving Average). В отличие от обычных скользящих, PMA использует линейную регрессию для прогнозирования будущих значений, уменьшая лаг.

Введите текст комментария

SBER

Формула

PMA:<code>PMA = SMA + Slope * Length / 2,

где Slope — наклон линии регрессии.

Дополнительно Элерс предложил прогнозировать саму PMA:
<code>PredictPMA = PMA + 0.5 (Slope - Slope[2]) Length

и наклон:
<code class="inline-code">PredictSlope = 1.5 Slope - 0.5 Slope[4].

Пересечения PredictPMA и PMA помогают находить точки входа и выхода, делая стратегию более адаптивной к изменениям рынка.

S&P 500 E-Mini Futures

Стратегия на основе индикатора PMA Джона Ф. Элерса

– Цена закрытия на дневном графике выше 50-дневной PMA.
– 10-дневная PMA выше 50-дневной.

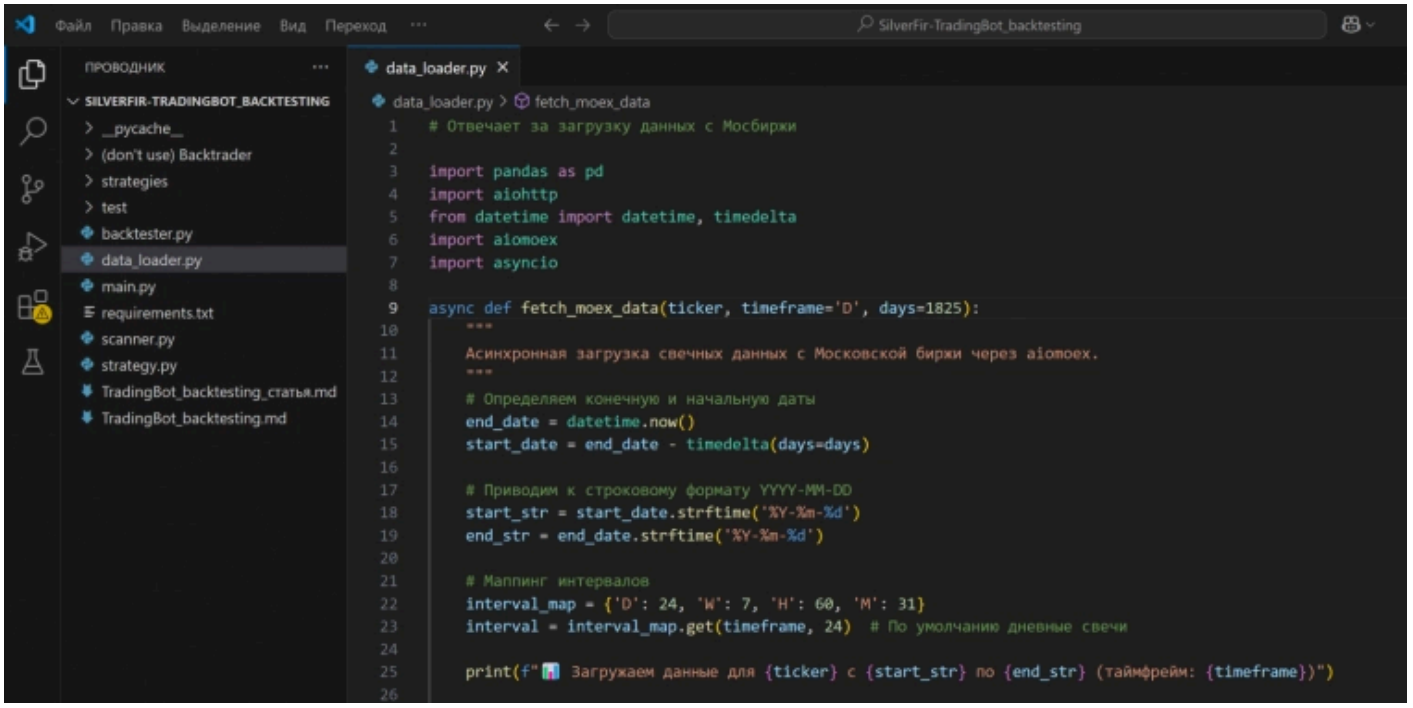
Риск-менеджмент:

– Первоначальный стоп-лосс устанавливается на 10% ниже цены входа.
– Выход из позиции осуществляется по скользящему стопу на основе ATR.

Реализация бэктестинга через backtesting.py. Определение топ-20 акций по объему

Весь код представлен на GitHub.

Модуль data_loader.py

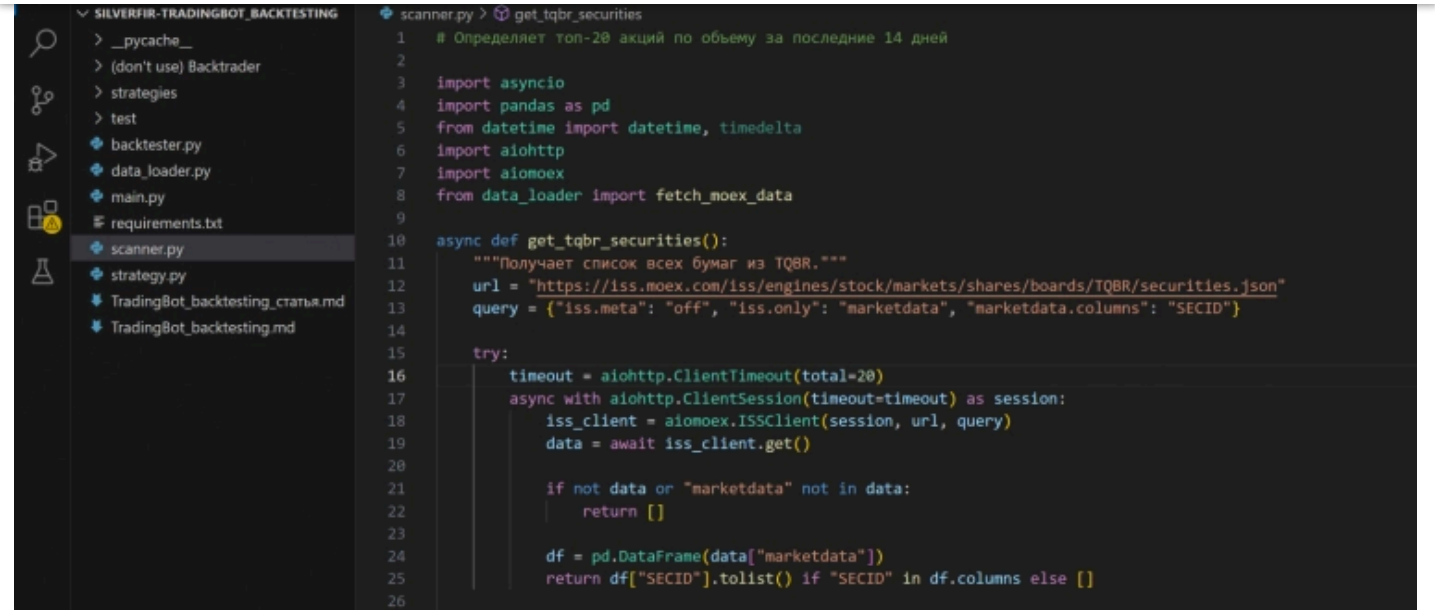


Для тестирования стратегии необходимо загружать актуальные данные о торгах.

В этом помогает библиотека `aiomoex`, которая предоставляет API-доступ к Московской бирже. В модуле `data_loader.py` реализована функция `fetch_moex_data`, позволяющая асинхронно получать исторические данные по свечам.

Функция запрашивает данные за последние 1825 дней (примерно 5 лет) и конвертирует их в формат Pandas DataFrame. Особенность реализации — использование асинхронного HTTP-клиента aiohttp, что ускоряет загрузку. Данные приводятся к удобному формату: преобразуются даты, устанавливается индекс, а названия колонок заменяются на стандартные для анализа.

Фильтрация ликвидных бумаг для тестирования. Модуль scanner.py



После загрузки данных важно отобрать ликвидные бумаги. Для этого в модуле `scanner.py` реализована функция get_top_20_stocks, которая анализирует объем торгов за последние 14 дней и выделяет 20 наиболее ликвидных акций.

Алгоритм работы следующий:

1. Получение списка всех торгуемых акций на основном рынке (TQBR) через API Московской биржи.
2. Асинхронная загрузка дневных данных по каждому инструменту с помощью fetch_moex_data.
3. Расчет суммарного объема торгов за 14 дней.
4. Формирование списка из 20 акций с наибольшим объемом.

Таким образом, отбираются бумаги с высоким оборотом, что повышает надежность тестирования стратегии и снижает риск торговли неликвидными активами.

Реализация бэктестинга через backtesting.py. Тестирование стратегии на исторических данных

Зачем всё разделил на модули?

Разделение кода на модули делает его более удобным для сопровождения, масштабирования и переиспользования. В нашем случае:

- data_loader.py отвечает за загрузку данных с Московской биржи.
- scanner.py фильтрует ликвидные бумаги.
- backtester.py выполняет бэктестинг.
- strategy.py содержит описание стратегии.

системы.

Пример кода для бэктестинга с использованием backtesting.py

```
<code>import asyncio
import pandas as pd
from backtesting import Backtest
from data_loader import fetch_moex_data
from strategy import LongOnlyPMAMultiTimeframeATRTrailingStop

async def run_backtest(ticker):
    print(f"\n{'='*50}")
    print(f"🚀 Запуск бэктеста для {ticker}")
    print(f"{'='*50}\n")

    # Получаем данные
    df, start_str, end_str = await fetch_moex_data(ticker) # Получаем start_str

    # Запускаем бэктест
    print("⏳ Запуск бэктеста...")
    strategy_class = LongOnlyPMAMultiTimeframeATRTrailingStop # Класс стратегии
    strategy_name = f"{ticker}_{start_str}_{end_str}_LongOnlyPMAMultiTimeframeAT
    DynamicStrategyClass = type(strategy_name, (strategy_class,), {}) # Создаем
    bt = Backtest(df, DynamicStrategyClass, cash=100_000, commission=0.002) # Ис
    stats = bt.run()

    # Вывод результатов
    print("\n📊 Результаты бэктеста:")
    print(f"⚙️ Стратегия: {strategy_name}") # Выводим динамическое имя стратеги
    print(f"🗓️ Период тестирования: с {stats['Start']} по {stats['End']}")
    print(f"💰 Начальный капитал: 100,000 руб.")
    print(f"💵 Конечный капитал: {stats['Equity Final [$]']:.2f} руб.")
    print(f"📈 Общая доходность: {stats['Return [%]']:.2f}%")
    print(f"📊 Годовая доходность: {stats['Return (Ann.) [%]']:.2f}%")
    print(f"📈 Коэффициент Шарпа: {stats['Sharpe Ratio']:.2f}")
    print(f"📉 Максимальная просадка: {stats['Max. Drawdown [%]']:.2f}%")
    print(f"🔄 Количество сделок: {stats['# Trades']}")
    print(f"✅ Процент выигрышных сделок: {stats['Win Rate [%]']:.2f}%")
```

```
    print("\n📊 Построение графика результатов...")
    try:
        bt.plot()
        print("✅ График успешно построен!")
    except ValueError as e:
        print(f"❌ Ошибка при построении графика: {e}")
    print(f"\n{'='*50}")
    print(f"🏁 Бэктест для {ticker} завершен")
    print(f"{'='*50}\n")
    return stats</code>
```

Основные метрики оценки

Для оценки стратегии используются ключевые метрики:

- Общая доходность (%) — показывает, сколько стратегия заработала за весь период тестирования.
- Годовая доходность (%) — усреднённый годовой прирост капитала.
- Коэффициент Шарпа — измеряет соотношение доходности к риску.
- Максимальная просадка (%) — определяет максимальную потерю капитала.
- Процент выигрышных сделок.
- Средняя продолжительность сделки.

Эти показатели позволяют оценить эффективность стратегии и принять решение о её использовании в реальной торговле.

Результаты тестирования на акциях Московской биржи

Как положительные, так и отрицательные.

Примеры в html файлах на GitHub'e.

СПБ Биржа (тикер SPBE):

Результаты бэктеста:

Стратегия: SPBE_2020-03-03_2025-03-02_LongOnlyPMAMultiTimeframeATRTrailingStop

Период тестирования: с 2021-11-19 00:00:00 по 2025-03-01 00:00:00

Начальный капитал: 100,000 руб.

Конечный капитал: 349138.97 руб.

Общая доходность: 249.14%

Годовая доходность: 47.34%

Коэффициент Шарпа: 0.80

Максимальная просадка: -27.41%

Количество сделок: 11

Процент выигрышных сделок: 54.55%

Лучшая сделка: +36.56%

Худшая сделка: -8.31%

Средняя продолжительность сделки: 25 days 00:00:00

Новатэк ао (тикер NVTK):

Результаты бэктеста:

Стратегия: NVTK_2020-03-03_2025-03-02_LongOnlyPMAMultiTimeframeATRTrailingStop

Период тестирования: с 2020-03-03 00:00:00 по 2025-03-01 00:00:00

Начальный капитал: 100,000 руб.

Конечный капитал: 94443.56 руб.

Общая доходность: -5.56%

Годовая доходность: -1.15%

Коэффициент Шарпа: -0.07

Максимальная просадка: -38.70%

Количество сделок: 22

Процент выигрышных сделок: 27.27%

Лучшая сделка: +18.55%

Худшая сделка: -10.78%

Средняя продолжительность сделки: 23 days 00:00:00

Проблема учета смены лидеров по объему

Мой код отдельно тестирует каждую акцию из топ-20 на момент отбора (на сегодня). Однако он не учитывает смену лидеров по объему и не позволяет работать с единой корзиной акций, где позиции могут удерживаться даже после выпадения бумаги из топ-20.

при которой состав портфеля регулярно пересматривается и обновляется на основе новых данных.

Следующие шаги

Фиксированный список топ-акций по объему устаревает. Использование динамического реестра позволит оперативно учитывать смену лидеров, корректируя состав активных позиций в стратегии.

Библиотека `ta-lib` мне не очень понравилась из-за сложностей с установкой — проще переписать индикатор вручную в будущем.

Получится ли реализовать это через `backtesting.py`? Скорее всего вряд ли.

Скорее всего придётся вернутся к `Backtrader`.

Заключение

Тестирование стратегии на акциях Мосбиржи показало её стабильную эффективность при использовании индикатора PMA на дневных свечах.

Python доказал свою ценность в алгоритмической торговле, обеспечивая гибкость и автоматизацию. Однако backtesting.py имеет ограничения.

Автор: Михаил Шардин

🔗 Моя онлайн-визитка

📢 Telegram «Умный Дом Инвестора»

11 марта 2025 г.

Python торговые роботы

3.6K 14 29 22

Михаил Шар…

Пермь

106 860

с 23 января 2019

+ Подписаться

homeinv

Не работают
backtesting.py => backtesting.ru?

0

Михаил Шардин
11 марта 2025, 06:47

> Хоббит, это ведь не ссылка… так библиотека называется
>
> Вот ссылка: https://kernc.github.io/backtesting.py/

+1

Никита Шляпников
11 марта 2025, 08:57

Решение запаздывания скользящий средней решается уменьшением таймфрейма, у Элиота в то время этой возможности не было, у нас есть.
15ma на часовом таймфрейме равна 900ma на минутном таймфрейме. Проблема решена)

0

SergeyJu
11 марта 2025, 09:13

Никита Шляпников, зачем 15 минут. Сама цена и есть неотстающий индикатор.

+1

Никита Шляпников
11 марта 2025, 09:20

SergeyJu, 15ma часового таймфрейма -> 15 moving avarange for timeframe 1 hours. Скользящие средние базис всех индикаторов и служит для принятия решения. Проблема большинства индикаторов строящихся на цене состоит в том что заранее не известно как закроется временной период (например часовик, при этом волатильность в течение часа может быть существенной) поэтому индикаторы на бектесте работают отлично, а в реальности ужасно, из-за эффекта запаздывания и несвоевременно принятого решения.

0

SergeyJu
11 марта 2025, 09:31

Никита Шляпников,
1. Скользяшки не базис, поскольку есть системы, в которых вообще не используются скользящки.

SergeyJu
11 марта 2025, 09:19

Не понял методологический посыл статьи.
Если тестируется индикатор, так надо на всех ликвидных в одном и том же временном окне его и бэктестить.
А если тестируется выбор сильнейшей акции по типу моментума, но на новом индикаторе, тоже проще было бы прогнать на фиксированном наборе акций, тем более, что список ликвидных за последние 5 лет не сильно изменился.

+2

OnlyHuman
11 марта 2025, 09:23

Спасибо за такой качественный и подробный пример. Сам потихоньку копаюсь в Python, ваш код мне будет очень полезен для дальнейшего погружения в эту тему.

+2

Михаил Шардин
11 марта 2025, 09:25

> OnlyHuman, на гитхабе выложил. Ещё туда залью — пока остановился немного.
> Для фьючерсов сделал что-то похожее только без библиотеки aiomoex.
> Всё таки проще без неё похоже работать напрямую с API Мосбиржи

+1

Riskplayer
11 марта 2025, 09:30

Если стоит Python в сборке от Anaconda, то установка ta-lib очень простая.

+1

SergeyJu
11 марта 2025, 09:41

Кстати, а угол наклона считается по ценам или по самой скользяшке и как связаны временное окно расчета линейной регрессии и параметр скользяшки.

0

Дмитрий Овчинников

> сервере, включая облачные решения и Raspberry Pi. "

Не уловил в чем именно торговое преимущество.

0

Напишите комментарий...

ОТПРАВИТЬ