ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
ФГАОУ ВО НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Образовательная программа «Прикладная математика и информатика»

Отчет о программном проекте

на тему: Торговая система для крипто-бирж

(промежуточный, этап 1)

Выполнили:

Студент группы БПМИ209		Л.В.Прокопчук
	Подпись	И.О.Фамилия
Студент группы БПМИ209		Л.И.Рыбаков
	Подпись	И.О.Фамилия
Студент группы БПМИ209		И.Ю.Бондырев
	Подпись	И.О.Фамилия

17.02.2022
Дата

Принял:

Руководитель проекта Казаков Евгений Александрович
Имя, Отчество, Фамилия

разработчик
Должность, ученое звание

Facebook inc.
Место работы (Компания или подразделение НИУ ВШЭ)

Дата проверки 17.02 2022 10
Оценка (по 10-ти бальной шкале) Подпись

Москва 2022

Содержание

1 Введение **2**
- 1.1 Актуальность проблемы . . . 2
- 1.2 Цели и задачи . . . 3
  - 1.2.1 Цель . . . 3
  - 1.2.2 Задачи . . . 3
- 1.3 Статьи . . . 3
- 1.4 Документация бирж . . . 3
- 1.5 Документация библиотек . . . 3
- 1.6 Аналоги . . . 3

2 Описание функциональных требований к программному проекту **4**
- 2.1 Сбор данных . . . 4
- 2.2 Измерение скорости соединения . . . 4
- 2.3 Коннектор к бирже . . . 4
- 2.4 Машинное обучение . . . 4
- 2.5 Коннектор . . . 4

3 Описание нефункциональных требований к программному проекту **5**
- 3.1 Безопасность . . . 5
- 3.2 Отказоустойчивость . . . 5
- 3.3 Скорость . . . 5
- 3.4 Переиспользование кода . . . 5
- 3.5 Масштабируемость . . . 5

4 Дополнительный результаты **5**
- 4.1 Визуализация . . . 5
- 4.2 Маркет-Мэйкинг стратегия . . . 6
- 4.3 Контролирующий бот . . . 6
- 4.4 CI/CD, тесты, линтер . . . 6
- 4.5 Смарт контракты . . . 6

Аннотация

Исследование и написание торговых стратегия для децентрализованных бирж на Layer-2[1].

1 Введение

1.1 Актуальность проблемы

Сегодня, кого не спроси, все знают, что такое биткоин, или, по крайней мере, говорят, что знают. Разговоры о криптовалютах и их производных в мире не утихают уже несколько лет, но, к сожалению, большинство из них очень поверхностные и не доходят даже до обсуждения принципа работы блокчейна в общих словах. Несмотря на кажущуюся сложность устройства, 2 самых больших блокчейна (Bitcoin и Ethereum) критически не справляются с нагрузкой, возложенной на них желающими воспользоваться их плюсами. Из-за того, что сеть Эфириума может обрабатывать лишь 15 транзакций в секунду, комиссия, которую нужно заплатить, чтобы транзакция была одной из этих 15 доходит до $70, что делает любой токен на блокчейне непригодным для использования в качестве обычной фиатной валюты. Чтобы снизить нагрузку на мэйннет[2], были разработаны и все еще разрабатываются несколько альтернативных решений, которые одним словом называются Layer-2 решения. Это надстройки над блокчейном, которые увеличивают пропускную способность и скорость в ущерб децентрализованности. Мы считаем, что пока не придумали более изящного способа достичь тех же

[1] Технологии масштабирования, позволяющие увеличить скорость и пропускную способность блокчейна. Обычно это делается за счет децентрализованности. Самые распространенные разновидности — rollups (свертки, собирают несколько транзакций в одну и записывают в блокчейн) и sidechain (блокчейн, который опирается на масштабируемую сеть. Часто используют менее децентрализованную и, как следствие, более быстрые консенсусные алгоритмы)

[2] Основная сеть блокчейна, на которой криптовалюта имеет реальную стоимость. Есть также сети для тестирования разработок. На них валюту можно получить по запросу от специальных адресов

результатов, которые дают L2 решения, данная технология будет развиваться, а актуальности нашей темы будет расти.

1.2 Цели и задачи

1.2.1 Цель

Написать трэйдинг систему, которая сможет стабильно выходить в плюс.

1.2.2 Задачи

- Проведение исследований по стратегиям трэйдинга.
- Проверка работоспособность стратегий.
- Создание инфраструктуры, позволяющей взаимодействовать с биржей автоматизированно.
- Сбор данных и обучение модели.
- Написание программы, совершающей сделки.
- Обзор и сравнительный анализ источников и аналогов

К сожалению, выбранная нами тема мало освещается в источниках любого вида: никто не захочет делиться прибыльной стратегией. Многое нам приходилось и придется делать с нуля.

1.3 Статьи

Тем не менее, существуют статьи, описывающие некоторые возможные подходы к написанию алгоритмов HFT. Например, есть ресурс [8], на котором описывается стратегия маркет-мейкинга, аналог которой мы попытались реализовать. Но материалы такого рода, находящиеся в открытом доступе, с течением времени теряют свою актуальность: если большое количество участников рынка придерживается одной схемы действий, то вскоре она перестает приносить прибыль. По этой причине мы старались не ориентироваться на подобные источники.

1.4 Документация бирж

Основным же источником информации для нас служила документация API [3] [1] бирж, к которым мы подключались. С помощью нее был написан коннектор, инкапсулирующий процесс подключения и взаимодействия с биржей, произведен сбор необходимой информации: список сделок за последний месяц, состояние о счете и т.п.

1.5 Документация библиотек

Для машинного обучения мы использовали CatBoost [2] — это библиотека от Яндекса для градиентного бустинга, надстройки над решающими деревьями. КэтБуст для нас лучшее решение, потому что это самая быстрая библиотека для классификации среди аналогов и проста в использовании.

1.6 Аналоги

На крипто валютном рынке существует множество торговых ботов, но информации об их характеристиках и принципах работы практически нет. Мы можем судить об их доходности, лишь по каким-то сомнительным заявлениям или косвенным признакам. В открытом доступе в основном находятся боты, которые предоставляют лишь интерфейс взаимодействия с биржей [7] [5]: “ручная” покупка и продажа токенов, выставление лимитных ордеров и т.п.. Такие решения не представляют для нас никакого интереса.

2 Описание функциональных требований к программному проекту

2.1 Сбор данных

Нужно обеспечить удобный механизм сбора исторических данных с бирж, на которых будет тестироваться и обучаться система.

2.2 Измерение скорости соединения

Так как счет идет на миллисекунды, мы всегда должны иметь четкое представления, какое время у нас займет отправка и получения пакета данных. Для этого должен быть предусмотрен отдельный модуль, который будет замерять скорость соединения с различными сервисами.

2.3 Коннектор к бирже

В программе должны быть коннекторы к биржам. Это класс, в конструктор которого подаются приватные ключи кошельков. После этого можно работать с биржей: смотреть информацию о счете, валютаю, отправлять и отменять ордера. Надо реализовать функционал, который предоставляет апи, чтобы можно было его использовать в трейдинг-стратегиях.

2.4 Машинное обучение

Индикаторов, по которым можно строить прогнозы, очень много, поэтому ручными методами не получится подобрать правильное соотношение весов. Здесь нужно машинное обучение, которое принимает на вход пред обработанные данные сделок, а на выход выдает модель, которую можно использовать в трейдинг-стратегии. Важно, чтобы модель была устойчива к тому, что в датасет добавляется или убирается индикаторы. Примеры использования

2.5 Коннектор

Предоставляем класс, который способен взаимодействовать с биржами по API, например:

- Отправка ордеров

```
connector.send_order(
        symbol=ETH_USD, side=BUY, price=1, quantity=0.1
)
```

- Получение текущих позиций

```
connector.get_our_positions(
        opened=True, symbol=ETH_USD
)
```

- Получение трейдов за определенный промежуток времени

```
connector.get_historical_trades(
        market=BTC_USD,
        begin="2021-12-12 09:00:00",
        end="2021-12-12 12:00:00"
)
```

- Информация о конкретном рынке

```
connector.get_symbol_info(market=ETH_USD)
```

- Измерение скорости

```
speed_measure = SpeedMeasure(connector)
speed_measure.get_connector_funcs_exec_times(
        market=ETH_USD,
        side=BUY,
        iters_num=10,
        filename="connector_funcs_exec_times.json",
    )
```

- Измерение задержки до биржи

```
speed_measure.get_orders_processing_delays(
        market=ETH_USD,
        side=BUY,
        orders_num=10,
        filename="orders_processing_delays.json",
)
```

3 Описание нефункциональных требований к программному проекту

3.1 Безопасность

Финансы — чувствительная тема, поэтому наша программа не должна допускать утечек данных о кошельках и приватных ключах. Нужно обеспечить безопасное хранение.

3.2 Отказоустойчивость

Во время трейдинга торговая система получает сотни обновлений от разных бирж, их обрабатывает, строит прогнозы и торгует. Нужно сделать так, чтобы система была готова к большим нагрузкам, и поведение было однозначно определено. Еще нужно проработать быстрое отключение торговой системы от торгов, если ее поведение станет неадекватным, и можно было бы быстро закрыть открытые заявки.

3.3 Скорость

В трейдинге важна каждая миллисекунда, поэтому цель — минимизировать время обработки, отправки и принятия данных.

3.4 Переиспользование кода

Нужно выстроить архитектуру проекта так, чтобы можно было быстро и легко тестировать свои гипотезы, поэтому код, который есть в проекте, должен быть написан так, чтобы его можно было легко понять и переиспользовать в других местах.

3.5 Масштабируемость

Система должна быть расширяема на несколько бирж и потенциально работать с большим количеством предсказательных моделей.

4 Дополнительный результаты

Помимо задач, которые нам поставил руководитель, мы сделали еще:

4.1 Визуализация

Перед тем, как работать с данными, надо понять, как они устроены: попарное распределение классов, плотность каждого из индикаторов. Мы все это сделали. Теперь стало гораздо проще подбирать параметры для модели машинного обучения.

4.2 Маркет-Мэйкинг стратегия

Это стратегия, когда мы держим окно из зеркальных заявок на каком-то уровне от индекс-цены. Утверждается, что если нас пробили с одной стороны, то почти сразу пробьют и со второй стороны, и мы окажемся в плюсе. Но все оказалось не так: при пробитии нас с одной стороны, рынок продолжал идти в ту же сторону.

4.3 Контролирующий бот

Мы сделали телеграм бота, на которого можно по паролю подписаться и получать обновления состояния аккаунта на бирже. Это полезно, когда ты запускаешь торговую стратегию, куда-то отходишь, но при этом всегда можешь контролировать, что с ней происходит, через телеграм.

4.4 CI/CD, тесты, линтер

Любая ошибка в торговой системе может потерять наши деньги, поэтому важно, чтобы весь код всегда был рабочим. Для этого мы все, что смогли, обложили тестами с использованием утилиты PyTest [11]. Теперь при каждом пулл реквесте у нас запускается тестирующая система, и если какие-то тесты не проходят, то мы запрещаем дальнейший пуш. Реализовали мы это через GitHub Actions [6].

Еще нам хочется, чтобы весь код был консистеным. Для этого мы используем линтер Black [9] и статический анализатор PyLint [10]. В совокупности эти утилиты поддерживают консистентность нашего кода и могут еще до запуска теста, выдать синтаксические ошибки.

4.5 Смарт контракты

С помощью смарт контрактов можно занимать у других людей миллиарды, трейдить на них, и потом возвращать криптовалюту обратно. Звучит заманчиво. Мы написали смарт конракты на Solidity [12], и залили их в сеть эфира через Brownie [4].

Список литературы

[1] Binance. Binance documentation. 2022. URL: https://binance-docs.github.io/apidocs/spot/en/#change-log.

[2] Catboost. Catboost documentation. 2022. URL: https://catboost.ai/en/docs/.

[3] DydxProtocol. Dydx documentation. 2022. URL: https://docs.dydx.exchange/#general.

[4] ETHBrownie. Brownie. 2022. URL: https://github.com/eth-brownie/brownie.

[5] FreqTrade. Free, open source crypto trading bot. 2022. URL: https://github.com/freqtrade/freqtrade.

[6] GitHub. Github actions. 2022. URL: https://docs.github.com/en/actions.

[7] Haehnchen. Cryptocurrency trading bot in javascript. 2022. URL: https://github.com/Haehnchen/crypto-trading-bot.

[8] HFTbattle. Hft strategy. 2022. URL: https://docs.hftbattle.com/ru/strategy/advanced.html.

[9] PFS. Black. 2022. URL: https://github.com/psf/black.

[10] PyCQA. Pylint. 2022. URL: https://github.com/PyCQA/pylint.

[11] PytestDev. Pytest. 2022. URL: https://github.com/pytest-dev/pytest.

[12] Solidity. Solidity. 2022. URL: https://docs.soliditylang.org/en/v0.8.12/.