Geekbrains

**Разработка телеграмм-бота, предназначенного для прогнозирования стоимости акций, облигаций, индексов и прочих финансовых инструментов, представленных на сайте finance.yahoo.com/markets/. Применение нейронных сетей различной архитектуры для решения данной задачи и выбор наилучшего результата на основе анализа выбранного финансового инструмента.**

Программа: Цифровые профессии

Инженер искусственный интеллект

Вепренцов Олег Михайлович

Железногорск

2024

**Основные требования к оформлению дипломного проекта**

1. Текст проекта оформляйте только с одной стороны листа формата А4. При наборе используйте межстрочный интервал 1,5.  
   Объемные таблицы/иллюстрации возможно размещать на листах А3, А2 и выносить в приложения.
2. При наборе информации, используйте шрифт Times New Roman с выравниванием текста по ширине. Рекомендуем кегль — 12–14, для некоторых структурных единиц работы можно выбрать полужирное выделение.
3. Поля страницы, где размещается текст, выставляйте по определенным показателям: нижнее/верхнее – 20 мм, правое – 10 мм, левое (для последующей прошивки) – 30 мм.
4. Все листы дипломного проекта нумеруются по порядку, начиная с введения (используется сквозная нумерация). Номер указывайте в центре нижней части страницы без точки.
5. Заголовок раздела с одним подразделом или пунктом не нумеруется.
6. После текста перед следующим заголовком установите одиночный интервал.
7. После заголовка перед подзаголовком проставьте двойной интервал.
8. Имена собственные — наименования компаний/изделий, фамилии учёных и т. д., указывайте на языке оригинала.
9. Каждый структурный элемент начинается с нового листа.

*Вся работа должна составлять 50+ страниц формата А4, включая приложения.*

|  |
| --- |
| 💡 Инструкция по работе с шаблоном.   1. Создайте копию данного шаблона. Файл - Создать копию. 2. Изучите шаблон и начинайте работу. |

**Содержание**

**Элементы оглавления не найдены.**

# Как оформить содержание?

В содержании дипломного проекта необходимо озаглавить основные блоки и подпункты работы с указанием страниц.

### Пример

*Данный пример не является эталонным, лишь один из вариаций.*

Введение (2–3 стр.)

Глава 1. Основы ручного тестирования веб-приложений (~15 стр.)

1.1 Что такое веб-приложение, его особенности

1.2 Что такое тестирование, кто такой тестировщик. Зачем нужно тестирование и когда оно начинается

1.3 Этапы ручного тестирования веб-приложений

1.4 Источники требований: что такое техническое задание и как проводить ручное тестирование, если техническое задание отсутствует

Глава 2. Подготовка к ручному тестированию веб-приложения (~15 стр)

2.1 Составление чек-листов

2.2 Составление тест-кейсов

2.3 Разработка тест-плана

2.4 Тест-дизайн и тест-аналитика

2.5 Баг-репорты и багтрекинговая система

Глава 3. Проведение тестирования и разработка предложений по улучшению тестирования Интернет-магазина РИВ ГОШ (~20 стр.)

3.1 Тестирование функциональности

3.2 Тестирование юзабилити

3.3 Тестирование интерфейса

3.4 Тестирование совместимости

3.5 Тестирование производительности

3.6 Тестирование безопасности

3.7 Подведение итогов. Разработка предложений по улучшению ручного тестирования веб-приложения РИВ ГОШ

Заключение (~ 4 стр.)

Список используемой литературы

Приложения

**Введение**

Задачей данного дипломного проекта является создание телеграмм бота для предсказания цены акций, облигаций, индексов и других финансовых инструментов сайта finance.yahoo.com/markets/.

Источником финансовых данных является сайт finance.yahoo.com, как очень крупный агрегатор данных о финансовых инструментах.

Для предсказания цены выбранного финансового инструмента используется несколько нейронных сетей разногой архитектуры, выбирается наиболее оптимальная и информация передается пользователю.

В качестве входных данных поступают только предыдущие значения выбранного символа (термин, используемый на сайте, означающий уникальный идентификатор финансового инструмента), без анализа новостного фона и значений других финансовых инструментов. Данное ограничение обусловлено невозможностью предоставить адекватные вычислительные мощности для целей дипломного проекта.

Несмотря на указанные ограничения, представляется возможным предоставление доступа к телеграмм-боту для телеграмм-каналов, специализирующихся на различных финансовых рынках, инвестировании и заработке. Бот может использоваться в качестве дополнительного инструмента для анализа при принятии инвестиционных решений. В случае роста популярности использования, возможно рассмотрение вопроса о монетизации.

При наличии финансирования для доступа к необходимым вычислительным ресурсам, существует возможность развития данного проекта с включением в качестве входных данных новостей, других связанных финансовых инструментов и иных показателей. Однако такое развитие выходит за рамки настоящего дипломного проекта.

Я лично давно интересуюсь инвестициями и имею некоторые знания в области финансовых инструментов, технического анализа и т.д., поэтому рассматриваю данный проект как полезный дополнительный инструмент для принятия решений по инвестициям.

В данном проекте будет применятся следующие технологии: Python, VSCode, JSON, библиотеки telegram, dotenv, requests, pandas, numpy, yfinance, sklearn, keras и pandas\_ta и другие.

Состав команды: Вепренцов Олег Михайлович, разработчик, тестировшик, технический писатель.

**1. Источник данных.**

Сайт finance.yahoo.com представляет собой финансовый портал, который предоставляет пользователям информацию о котировках акций, облигаций, валютных пар и других финансовых инструментов.

Основная цель сайта — предоставление актуальной информации о состоянии финансовых рынков. Для этого на сайте есть разделы, где пользователи могут отслеживать изменение цен на акции, курсы валют, графики движения цен и другие данные.

На сайте представлены следующие разделы:

1. Акции: Пользователи могут отслеживать изменение цен на акции компаний различных отраслей и стран. Информация включает в себя текущие цены, графики движения цен, исторические данные и прогнозы аналитиков.
2. Облигации: На сайте представлена информация о ценах на государственные и корпоративные облигации, а также о доходности этих инструментов.
3. Валютные пары: Пользователи могут следить за курсами основных мировых валют, включая доллар США, евро, японскую йену и другие.
4. Индексы: На сайте представлены данные о динамике основных фондовых индексов, таких как S&P 500, Dow Jones Industrial Average, NASDAQ Composite и другие.
5. Сырьевые товары: Пользователи могут получить информацию о ценах на нефть, золото, серебро и другие сырьевые товары.
6. Криптовалюты: На сайте представлены котировки основных криптовалют, таких как Bitcoin, Ethereum, Litecoin и другие.

Список доступных символов можно посмотреть в приложении №1.

Сайт также предлагает инструменты для анализа финансовых данных, такие как калькуляторы доходности инвестиций, графики изменения цен и прогнозы аналитиков.

Кроме того, на сайте есть раздел новостей, где публикуются последние новости из мира финансов и экономики.

Таким образом, сайт finance.yahoo.com является полезным инструментом для инвесторов и трейдеров, позволяющим им следить за изменениями на финансовых рынках и принимать обоснованные решения при выборе инвестиционных стратегий.

**2. Телеграмм-бот в качестве интерфейса пользователя.**

Телеграмм боты представляют собой программные приложения, которые могут выполнять различные задачи в рамках мессенджера Telegram. Они могут быть использованы для автоматизации процессов, предоставления информации, выполнения определенных функций и многого другого. Боты могут быть созданы для различных целей, включая поддержку клиентов, продажи товаров и услуг, управление проектами и многое другое.

Одним из главных преимуществ телеграмм ботов является их удобство использования. Пользователи могут взаимодействовать с ботом через простой интерфейс, который обычно включает в себя текстовые сообщения и кнопки. Это делает процесс общения с ботом интуитивно понятным и легким для освоения. Кроме того, боты могут работать круглосуточно, что позволяет пользователям получать информацию и услуги в любое время дня и ночи.

Также стоит отметить, что телеграмм боты могут быть интегрированы с другими системами и сервисами, что расширяет их функциональность. Например, бот может быть подключен к базе данных или CRM-системе для получения информации о клиентах или заказах. Это позволяет создавать более сложные и эффективные решения для бизнеса.

Однако, несмотря на все преимущества, у телеграмм ботов есть и недостатки. Один из них – ограниченность функционала. Хотя боты могут выполнять множество задач, они все же ограничены в своих возможностях по сравнению с полноценными веб-приложениями или мобильными приложениями. Кроме того, взаимодействие с ботом может быть менее естественным, чем общение с человеком, особенно когда речь идет о сложных вопросах или проблемах.

Телеграмм боты являются эффективным инструментом для взаимодействия с пользователями, но они имеют свои ограничения. Для улучшения качества взаимодействия необходимо продолжать работу над улучшением функционала ботов, а также разрабатывать новые способы интеграции с другими системами и сервисами.

Телеграмм-бот был выбран в качестве интерфейса взаимодействия с пользователем в данном проекта в силу простоты реализации и удобства работы. Также мессенджер Telegram является очень популярным.

**3. Нейронная сеть для анализа данных.**

Нейронные сети представляют собой математические модели, которые имитируют структуру и функции биологических нейронов. Они используются для обработки и анализа данных, а также для решения сложных задач, таких как классификация, распознавание образов и прогнозирование. Нейронные сети могут быть обучены на основе большого количества данных, что позволяет им адаптироваться к новым условиям и решать задачи, которые не могут быть решены традиционными методами.

Одним из главных преимуществ нейронных сетей является их способность обрабатывать большие объемы данных и находить скрытые закономерности. Они могут обрабатывать данные в реальном времени, что делает их идеальными для анализа больших объемов данных. Кроме того, нейронные сети могут обучаться на основе новых данных, что позволяет им постоянно улучшать свою производительность.

Однако, несмотря на все преимущества, у нейронных сетей есть и недостатки. Один из них – сложность интерпретации результатов. Нейронная сеть может дать правильный ответ, но не всегда понятно, почему она приняла именно такое решение. Это может быть проблемой, особенно когда требуется объяснение причин принятия решения.

Нейронные сети являются эффективным инструментом для анализа данных, но они имеют свои ограничения. Для улучшения качества анализа необходимо продолжать работу над улучшением алгоритмов обучения нейронных сетей, а также разрабатывать новые методы интерпретации результатов.

Нейронные сети были выбраны для анализа финансовых данных в данном проекте так как я обучаюсь по направлению «Искусственный интеллект», и основной задачей дипломного проекта является показать знания и навыки, которые были получены мной в процессе обучения по данному направлению в компании GeekBrains. Использование нейронных сетей позволит в дальнейшем развивать проект при необходимости добавляя для анализа другие данные (новости, индексы, статистические данные и т.п.)

Также для работы нейронных сетей были использованы вспомогательные инструменты технического анализа, а именно несколько технических индикаторов, такие как RSI и скользящие средние. Опишем их более подробно ниже.

Индикатор RSI (Relative Strength Index) - это технический индикатор, который используется для измерения относительной силы ценового движения актива. Он основан на сравнении средних цен закрытия за определенный период времени. RSI колеблется между значениями от 0 до 100. Значения ниже 30 обычно считаются зоной перепроданности, а значения выше 70 - зоной перекупленности. Когда RSI достигает этих зон, это может сигнализировать о возможном развороте тренда.

Скользящие средние (Moving Averages, MA) - это еще один популярный технический индикатор, который используется для сглаживания ценовых колебаний и выявления трендов. Скользящая средняя рассчитывается путем усреднения цен закрытия за определенный период времени. Чем больше период, тем более сглаженной будет линия MA. Различные типы скользящих средних могут использоваться для разных целей, например, простая скользящая средняя (Simple Moving Average, SMA) или экспоненциальная скользящая средняя (Exponential Moving Average, EMA).

Скользящие средние могут использоваться для определения направления тренда, поддержки и сопротивления, а также для выявления дивергенций и конвергенций. Когда цена пересекает скользящую среднюю снизу вверх, это может сигнализировать о начале восходящего тренда, а когда цена пересекает скользящую среднюю сверху вниз - о начале нисходящего тренда.

Оба этих индикатора могут быть использованы вместе для анализа рынка и принятия торговых решений. Например, если RSI находится в зоне перекупленности, а цена пересекает скользящую среднюю сверху вниз, это может быть сигналом к продаже.

В проекте используются нейронные сети архитектуры LSTM (Long Short-Term Memory) - это разновидность рекуррентной нейронной сети, которая была разработана для решения проблемы долговременной памяти в нейронных сетях. Это позволяет нейронной сети запоминать информацию на протяжении длительного периода времени, что делает её особенно полезной для задач, где требуется учитывать долгосрочные зависимости.

LSTM состоит из нескольких слоёв, каждый из которых содержит ячейки LSTM. Ячейка LSTM имеет три компонента: входной вентиль, забывающий вентиль и выходной вентиль. Входной вентиль определяет, какую информацию следует сохранить, забывающий вентиль решает, какую информацию следует забыть, а выходной вентиль определяет, какая информация должна быть использована для формирования выходного сигнала.

LSTM работает следующим образом: на каждом шаге временного ряда входные данные подаются на вход ячейки LSTM, где они проходят через входной вентиль. Затем забывающий вентиль определяет, какие части информации должны быть забыты, и пропускает только те части, которые важны для текущего шага. Наконец, выходной вентиль формирует выходной сигнал, который используется для обучения модели.

LSTM широко применяется в различных областях, включая обработку естественного языка, распознавание речи, машинный перевод, прогнозирование временных рядов и многое другое.

Выбор LSTM в данном проекте обусловлен необходимостью анализа временных рядов, которыми являются финансовые данные. При дальнейшем развитии проекта возможно использование и других архитектур нейронных сетей.

**4. Архитектура приложения.**

Приложение имеет простую архитектуру. Спроектировано таким образом, чтобы была возможность легко расширить функционал, а также внести изменения в отдельные части приложения.

Основа – телеграмм бот, написанный с использованием библиотеки telegram, остальной функционал вынесен в отдельные модули, которые вызываются при выполнении команд. Он состоит из основных и сервисных модулей. Основными являются модуль сбора данных и собственно выдачи прогноза на основе нейронных сетей. Сервисные модули, это хранение данных и логирование действий пользователей.

Исходный код проекта находится по адресу <https://github.com/odolod/FinTechTB>

Телеграмм бот находится в файлах bot\_commands.py и telbot.py. Файл telbot.py является основным файлом запуска проекта.

Основной модуль сбора данных находится в файле finance\_yahoo\_data.py

Основной модуль прогноза находится в файле neuro.py

Сервисные модули находятся в файлах database.py и log.py

Также в каталоге приложения есть файлы \*.ipynb они не используются в приложении, но в данных блокнотах проводилась работа по подготовке данных и разработке моделей. Проверка и тестирование. Затем основной код из блокнотов переносился в приложение.

**4.1. Телеграмм бот.**

Телеграмм бот находится в файлах bot\_commands.py и telbot.py. Опишем код этих файлов более подробно. Сам исходный код можно посмотреть по адресу <https://github.com/odolod/FinTechTB> и в приложении №2.

Файл bot\_commands.py представляет собой часть реализации телеграмм бота, который предназначен для предсказания движения финансовых рынков. Бот предоставляет ряд команд, которые позволяют пользователю настраивать параметры, такие как интервал и символ финансового инструмента, а также загружать и просматривать данные.

В начале кода импортируются необходимые модули и классы, включая Update и ContextTypes из библиотеки telegram.ext, а также модули для логирования, работы с базой данных и получения данных с сайта finance.yahoo.com.

Далее следуют определения асинхронных методов, каждый из которых отвечает за обработку определенной команды. Метод hello\_command отвечает на команду /hello, отправляя приветственное сообщение пользователю. Метод start\_command отвечает на команду /start, предоставляя список доступных команд. Метод set\_interval\_command устанавливает интервал для отображения данных, а метод set\_symbol\_command устанавливает символ финансового инструмента. Метод settings\_command отображает текущие настройки пользователя, а метод load\_command загружает данные для выбранного символа и интервала. Метод prediction\_command выдает пользователю предсказание поведения выбранного финансового инструмента.

Каждый метод принимает два аргумента: update, который представляет собой объект обновления, и context, который представляет контекст текущего вызова. Методы используют эти аргументы для ответа пользователю и записи данных в базу.

Телеграмм бот запускается в файле telbot.py:

Сначала код проверяет наличие файла .env в текущем каталоге и загружает его содержимое, если он существует. Файл .env содержит переменные окружения, включая токен телеграмм бота (TGBOT\_TOKEN), который необходим для авторизации бота.

Затем создается экземпляр класса ApplicationBuilder и ему передается токен бота. После этого вызывается метод build(), который возвращает объект Application, представляющий собой основной класс для управления ботом.

Далее в коде добавляются обработчики команд, которые будут реагировать на определенные команды, отправленные пользователем. Каждый обработчик команды создается с помощью класса CommandHandler и регистрируется в приложении с помощью метода add\_handler(). Команды, которые обрабатываются в этом примере, включают hello, start, setInterval, setSymbol, settings, load и prediction.

Последний шаг - запуск процесса опроса, который осуществляется вызовом метода run\_polling() объекта Application. Этот метод запускает цикл опроса, который ожидает входящих сообщений и обрабатывает их в соответствии с зарегистрированными обработчиками команд.

Подключится к тестовому телеграмм боту можно по ссылке: <https://t.me/Test_547_bot>. Он будет работать примерно неделю после сдачи дипломного проекта, возможно с кратковременными перерывами.

**4.2. Сервисные модули.**

Модуль логирования находится в файле log.py и представляет собой часть телеграмм бота, которая используется для ведения журнала активности бота в формате CSV. Функция принимает два аргумента: update, который представляет собой объект обновления, и command, который указывает на команду, которую пользователь отправил боту.

Функция открывает файл logdb.csv в режиме добавления ('a'), затем записывает строку в файл. Строка состоит из следующих значений: название команды, текущее время, идентификатор пользователя, имя пользователя и текст сообщения. Каждое значение отделено запятой. После записи строки функция закрывает файл.

Сервисный модуль хранения данных реализован в файле database.py и представляет собой набор функций для работы с файлами JSON, которые содержат настройки пользователя и данные финансовых инструментов. Код использует библиотеки json и pathlib для чтения и записи данных в файлы.

Функция read\_json читает данные из файла JSON и возвращает список объектов. Функция write\_json записывает список объектов в файл JSON.

Функции save\_user\_interval, save\_user\_symbol, save\_chart\_data и load\_chart\_data используются для сохранения и загрузки данных пользователя и данных финансовых инструментов.

Функция load\_user\_settings загружает настройки пользователя из файла JSON.

В проекте реализовано сохранение данных полученных с сайта <https://finance.yahoo.com/>, при развитии проекта планируется замена файла JSON на систему управления базами данных, предварительно рассматривается MongoDB. Также планируется использование этих данных для визуализации графиков, и более глубокого анализа с использование нейронных сетей. Также в случае наличия данных в базе, не использовать запрос к finance.yahoo.com, а использовать уже сохраненные данные, сейчас при каждом запросе прогноза запрашиваются данные с сайта, но это некритично в рамках дипломного проекта.

Хранение настроек пользователя реализовано в файле settings.json, данный файл

является JSON-документом, который описывает список пользователей. Каждый объект списка содержит четыре поля:

1. UserID - числовой идентификатор пользователя.
2. UserName - строковое имя пользователя.
3. symbol - строковый тикер финансового инструмента.
4. interval - строковый интервал для запроса данных.

Объекты в списке могут иметь различное количество полей, в зависимости от того, какие данные были сохранены для каждого пользователя.

**4.3. Модуль загрузки данных.**

Модуль загрузки данных с сайта <https://finance.yahoo.com/> находится в файле finance\_yahoo\_data.py и представляет собой набор функций для получения данных с сайта finance.yahoo.com. Код использует библиотеку requests для отправки HTTP-запросов имитации поведения браузера при взаимодействии с сайтом.

Функция get\_yahoo\_cookie получает cookie с сайта finance.yahoo.com, который необходим для доступа к защищенным ресурсам. Функция get\_yahoo\_crumb получает crumb, который используется для предотвращения спама и DDoS-атак.

Функция get\_yahoo\_quote запрашивает информацию о указанному символу. Функция get\_yahoo\_symbol\_name получает название компании по ее символу. Функция get\_yahoo\_chart запрашивает финансовые данные по символу и интервалу, период времени выбран максимальный, он зависит от выбранного интервала и символа.

Доступ к данным осуществляется посредством API запросов, сложность реализации обусловлена тем, что API сайта является закрытым внутренним инструментом, для передачи данных между различными частями портала yahoo.com. Таким образом нет официального описания данного API и не предполагается доступ к нему сторонних разработчиков. Для доступа пришлось использовать анализ веб-страниц портала yahoo.com в режиме разработчика.

Благодаря подсказке преподавателя GeekBrains Тимура Исламгулова, огромное ему спасибо! Было решено перейти на библиотеку yfinance (<https://github.com/ranaroussi/yfinance>). Загрузка данных в модуле предсказания реализована через данную библиотеку. Модуль загрузки на requests был оставлен в проекте просто для демонстрации проделанной работы и фактически не используется для предсказания цены.

**4.3. Модуль предсказания.**

Модуль предсказания цены на основе нейронных сетей находится в файле neuro.py и представляет собой набор функций для прогнозирования цен финансовых активов с использованием LSTM (Long Short-Term Memory) нейронной сетей. Код использует библиотеки pandas, numpy, yfinance, sklearn, keras и pandas\_ta для загрузки исторических данных, обработки и обучения модели.

Функция get\_period\_from\_interval выбирает соответствующий период для заданного интервала. Функция load\_data загружает исторические данные по акции с помощью библиотеки yfinance.

Функции predict\_lstm и predict\_ta\_lstm подготавливают данные для обучения модели и предсказывают будущие цены. Функция predict\_ta\_lstm дополнительно включает технические индикаторы, такие как RSI и скользящие средние, для улучшения прогнозирования.

**5. Заключение.**

В ходе выполнения дипломного проекта был разработан телеграмм-бот для предсказания цены финансовых инструментов на основе исторических данных с сайта finance.yahoo.com. Использование нейронных сетей позволило достичь точности предсказания, достаточной для использования в качестве дополнительного инструмента при принятии инвестиционных решений. Были использованы современные технологии и методы анализа данных, включая нейронные сети и технические индикаторы

Разработанный продукт может быть интегрирован в телеграмм-каналы, специализирующиеся на финансовых рынках, и использоваться в качестве дополнительного инструмента для анализа при принятии инвестиционных решений.

Проект показал потенциал для дальнейшего развития и расширения функционала, включая интеграцию с другими источниками данных и использование дополнительных методов анализа. Также была отмечена возможность монетизации проекта в будущем, что может стать дополнительным стимулом для его развития и улучшения.

В целом, данный диплом подтверждает компетентность автора в области разработки программного обеспечения и применения искусственного интеллекта для решения практических задач.

**5. Список используемой литературы.**

● Сайт: [Прогнозирование временных рядов](https://www.tensorflow.org/tutorials/structured_data/time_series?hl=ru#part_1_forecast_a_univariate_time_series).

● Сайт: [Stock Market Analysis + Prediction using LSTM](https://www.kaggle.com/code/faressayah/stock-market-analysis-prediction-using-lstm/notebook).

● Сайт: [Пишем диалоговые Telegram-боты на Питоне](https://habr.com/ru/articles/316666/).

● Сайт: [Индикатор Rsi (Relative Strength index)](https://smart-lab.ru/company/os_engine/blog/983334.php).

● Сайт: [Download market data from Yahoo! Finance's API](https://pypi.org/project/yfinance/).

● Сайт: <https://finance.yahoo.com/markets/>.

● Сайт:

**6.1. Приложение №1 список символов.**

В данном приложении планировалось разместить таблицу символов, которые можно указывать для анализа, но оказалось, что их более чем 300000 на Yahoo! Finance, если указать их все, то это слишком сильно увеличит объём диплома, их можно поискать по [ссылке](https://finance.yahoo.com/lookup), укажем только наиболее популярные:

* 1. **Приложение №2 исходные коды.**

Файл telbot.py:

from telegram.ext import ApplicationBuilder, CommandHandler

from bot\_commands import \*

import os

from dotenv import load\_dotenv

#Токен телеграмм бота

dotenv\_path = os.path.join(os.path.dirname(\_\_file\_\_), ".env")

if os.path.exists(dotenv\_path):

    load\_dotenv(dotenv\_path)

token = os.getenv("TGBOT\_TOKEN")

app = ApplicationBuilder().token(token).build()

app.add\_handler(CommandHandler("hello", hello\_command))

app.add\_handler(CommandHandler("start", start\_command))

app.add\_handler(CommandHandler("setInterval", set\_interval\_command))

app.add\_handler(CommandHandler("setSymbol", set\_symbol\_command))

app.add\_handler(CommandHandler("settings", settings\_command))

app.add\_handler(CommandHandler("load", load\_command))

app.add\_handler(CommandHandler("prediction", prediction\_command))

app.run\_polling()

Файл log.py:

from telegram import Update

from datetime import datetime

def log(update: Update, command) -> None:

    file = open('logdb.csv','a')

    file.write(f'{command}, {datetime.now()}, {update.effective\_user.id}, {update.effective\_user.first\_name}, {update.message.text}\n')

Файл bot\_commands.py:

from telegram import Update

from telegram.ext import ContextTypes

from log import log

import pandas as pd

from database import save\_user\_symbol, save\_user\_interval, load\_user\_settings, load\_chart\_data, save\_chart\_data

from finance\_yahoo\_data import get\_yahoo\_symbol\_name, get\_yahoo\_chart

from neuro import load\_data, predict\_ta\_lstm

async def hello\_command(update: Update, context: ContextTypes.DEFAULT\_TYPE) -> None:

    log(update, 'hello\_command')

    await update.message.reply\_text(f'Привет {update.effective\_user.first\_name} это бот для предсказания движения финансовых рынков')

async def start\_command(update: Update, context: ContextTypes.DEFAULT\_TYPE) -> None:

    log(update, 'start\_command')

    await update.message.reply\_text(f'Команды:\n/hello\n/setInterval\n/setSymbol\n/settings\n/load\n/prediction\n/start')

async def set\_interval\_command(update: Update, context: ContextTypes.DEFAULT\_TYPE) -> None:

    log(update, 'set\_interval\_command')

    items = update.message.text.split()

    interval = "1d" # интервал

    valid\_interval = ["1d","5d","1mo","3mo"]

    try:

        interval = items[1] # интервал

    except Exception:

        await update.message.reply\_text(f'Интервал не задан, берем по умолчанию 1d')

    else:

        if interval in valid\_interval:

            await update.message.reply\_text(f'Интервал {interval} установлен')

        else:

            await update.message.reply\_text(f'Интервал {interval} неверный, допустимые: 1d, 5d, 1mo, 3mo, установлен 1d')

            interval = "1d"

    finally:

        #Записываем данные интервала для пользователя.

        save\_user\_interval(update.effective\_user.id, update.effective\_user.first\_name, interval)

async def set\_symbol\_command(update: Update, context: ContextTypes.DEFAULT\_TYPE) -> None:

    log(update, 'set\_symbol\_command')

    items = update.message.text.split()

    symbol = "TSLA" # тикер на https://finance.yahoo.com/trending-tickers

    try:

        symbol = items[1] # тикер

    except Exception:

        await update.message.reply\_text(f'Тикер не задан, берем по умолчанию TSLA')

    else:

        name = get\_yahoo\_symbol\_name(symbol)

        if name:

            await update.message.reply\_text(f'Тикер {symbol} ({name}) установлен')

        else:

            await update.message.reply\_text(f'Тикер {symbol} неверный, допустимые: https://finance.yahoo.com/trending-tickers, установлен TSLA')

            symbol = "TSLA"

    finally:

        #Записываем данные тикера для пользователя.

        save\_user\_symbol(update.effective\_user.id, update.effective\_user.first\_name, symbol)

async def settings\_command(update: Update, context: ContextTypes.DEFAULT\_TYPE) -> None:

    log(update, 'settings\_command')

    settings = load\_user\_settings(update.effective\_user.id)

    if settings:

        await update.message.reply\_text(f'Тикер:{settings["symbol"]} Интервал:{settings["interval"]}')

    else:

        await update.message.reply\_text(f'Настройки не найдены. Установите:/setInterval\n/setSymbol')

async def load\_command(update: Update, context: ContextTypes.DEFAULT\_TYPE) -> None:

    log(update, 'load\_command')

    settings = load\_user\_settings(update.effective\_user.id)

    if settings:

        symbol = settings["symbol"]

        interval = settings["interval"]

        if symbol and interval:

            chart = get\_yahoo\_chart(symbol,interval)

            if chart:

                save\_chart\_data(symbol,interval,chart)

                await update.message.reply\_text(f'Загрузка успешна.')

            else:

               await update.message.reply\_text(f'Загрузка не удалась.')

        else:

            await update.message.reply\_text(f'Проверьте настройки. Загрузка невозможна.')

    else:

        await update.message.reply\_text(f'Hастройки не найдены. Загрузка невозможна.')

async def prediction\_command(update: Update, context: ContextTypes.DEFAULT\_TYPE) -> None:

    log(update, 'prediction\_command')

    settings = load\_user\_settings(update.effective\_user.id)

    if settings:

        symbol = settings["symbol"]

        interval = settings["interval"]

        if symbol and interval:

            df = load\_data(symbol, interval)

            await update.message.reply\_text(f'Настройки найдены. просим подождать предсказания...')

            name = get\_yahoo\_symbol\_name(symbol)

            price = predict\_ta\_lstm(df)

            await update.message.reply\_text(f'Предсказанная стоимость {name} на следудующий интервал составит: {price[0][0]}')

        else:

            await update.message.reply\_text(f'Проверьте настройки. Не получилось предсказать.')

    else:

        await update.message.reply\_text(f'Hастройки не найдены. Предсказание невозможно.')

Файл database.py:

import json

from pathlib import Path

# [{UserID, UserName, Interval, Symbol}]

settings\_file = 'settings.json'

# [{symbol, interval, [chartData]}]

data\_file = 'charts.json'

def read\_json(filename: str) -> list:

    settigs = []

    with open(Path.cwd() / filename, 'r', encoding='utf-8') as fin:

        settigs = json.load(fin)

    return settigs

def write\_json(filename: str,settigs: list):

    with open(Path.cwd() / filename, 'w', encoding='utf-8') as fout:

        fout.write(json.dumps(settigs) + '\n')

def save\_user\_interval(user\_id, user\_name, interval):

    settings = read\_json(settings\_file)

    user = {}

    for set in settings:

        if set["UserID"] == user\_id:

            user = set

    if user:

        user["interval"] = interval

    else:

        settings.append({'UserID':user\_id, 'UserName':user\_name, 'interval':interval})

    write\_json(settings\_file,settings)

def save\_user\_symbol(user\_id, user\_name, symbol):

    settings = read\_json(settings\_file)

    user = {}

    for set in settings:

        if set["UserID"] == user\_id:

            user = set

    if user:

        user["symbol"] = symbol

    else:

        settings.append({'UserID':user\_id, 'UserName':user\_name, 'symbol':symbol})

    write\_json(settings\_file,settings)

def save\_chart\_data(symbol, interval, chart\_data):

    charts = read\_json(data\_file)

    symbol\_data = {}

    for chart in charts:

        if chart["symbol"] == symbol and chart["interval"] == interval:

            symbol\_data = chart

    if symbol\_data:

        symbol\_data["chartData"] = chart\_data

    else:

        charts.append({'symbol':symbol, 'interval':interval, 'chartData':chart\_data})

    write\_json(data\_file,charts)

def load\_chart\_data(symbol, interval):

    charts = read\_json(data\_file)

    symbol\_data = {}

    for chart in charts:

        if chart["symbol"] == symbol and chart["interval"] == interval:

            symbol\_data = chart

    return symbol\_data

def load\_user\_settings(user\_id):

    settings = read\_json(settings\_file)

    user = {}

    for set in settings:

        if set["UserID"] == user\_id:

            user = set

    return user

Файл finance\_yahoo\_data.py:

import requests

from fake\_useragent import UserAgent

import json

headers = {"User-Agent": UserAgent().chrome}

cookie = None

crumb = None

def get\_yahoo\_cookie():

    cookie = None

    response = requests.get(

        "https://fc.yahoo.com", headers=headers, allow\_redirects=True

    )

    if not response.cookies:

        raise Exception("Failed to obtain Yahoo auth cookie.")

    cookie = list(response.cookies)[0]

    return cookie

def get\_yahoo\_crumb(cookie):

    crumb = None

    crumb\_response = requests.get(

        "https://query1.finance.yahoo.com/v1/test/getcrumb",

        headers=headers,

        cookies={cookie.name: cookie.value},

        allow\_redirects=True,

    )

    crumb = crumb\_response.text

    if crumb is None:

        raise Exception("Failed to retrieve Yahoo crumb.")

    return crumb

def check\_crumb():

    global crumb

    global cookie

    if not crumb:

        cookie = get\_yahoo\_cookie()

        crumb = get\_yahoo\_crumb(cookie)

#Запрос данных по символу

#https://query2.finance.yahoo.com/v7/finance/quote?symbols=TSLA&crumb=[crumb-value]

def get\_yahoo\_quote(symbol):

    check\_crumb()

    params = {

        "symbols": symbol,

        "crumb": crumb,

    }

    quote\_response = requests.get(

        "https://query2.finance.yahoo.com/v7/finance/quote",

        headers=headers,

        cookies={cookie.name: cookie.value},

        params=params,

        allow\_redirects=True,

    )

    quote = quote\_response.text

    if quote is None:

        raise Exception("Failed to retrieve Yahoo quote.")

    return quote

def get\_yahoo\_symbol\_name(symbol):

    quote = get\_yahoo\_quote(symbol)

    name = None

    try:

        data = json.loads(quote)

        name = data["quoteResponse"]["result"][0]["shortName"]

    except Exception:

        name = None

    return name

#https://query1.finance.yahoo.com/v8/finance/chart/GOOGL?symbol=AAPL&period1=0&period2=9999999999&interval=1d

#"validRanges":["1d","5d","1mo","3mo","6mo","1y","2y","5y","10y","ytd","max"]}

def get\_yahoo\_chart(symbol, interval):

    check\_crumb()

    params = {

        "crumb": crumb,

        "interval": interval,

        "period1": 0,

        "period2": 9999999999

    }

    chart\_response = requests.get(

        "https://query1.finance.yahoo.com/v8/finance/chart/"+symbol,

        headers=headers,

        cookies={cookie.name: cookie.value},

        params=params,

        allow\_redirects=True,

    )

    chart = chart\_response.text

    if chart is None:

        raise Exception("Failed to retrieve Yahoo quote.")

    return chart

Файл neuro.py:

import pandas as pd

import numpy as np

import yfinance as yf

from sklearn.preprocessing import MinMaxScaler

from keras.models import Sequential

from keras.layers import Dense, LSTM

from keras.metrics import MeanAbsoluteError

import pandas\_ta as ta

import datetime as dt

def get\_period\_from\_interval(interval):

    period = "1y"

    if interval == '1mo' : period = '5y'

    if interval == '3mo' : period = '10y'

    if interval == '5d' : period = '2y'

    return period

def load\_data(symbol, interval):

    df = yf.Ticker(symbol).history(interval=interval, period=get\_period\_from\_interval(interval))

    return df

def predict\_lstm(df):

    # подготовка данных

    # новый фрейм данных только с столбцом Close.

    data = df.filter(['Close'])

    # преобразуем в Numpy array

    dataset = data.values

    # количество строк train

    training\_data\_len = int(np.ceil( len(dataset) \* .95 ))

    print(training\_data\_len)

    # маштабируем данные

    scaler = MinMaxScaler(feature\_range=(0,1))

    scaled\_data = scaler.fit\_transform(dataset)

    # создаем и масштабируем трэйн

    train\_data = scaled\_data[0:int(training\_data\_len), :]

    # Разделим данные на наборы данных x\_train и y\_train.

    x\_train = []

    y\_train = []

    for i in range(60, len(train\_data)):

        x\_train.append(train\_data[i-60:i, 0])

        y\_train.append(train\_data[i, 0])

        if i<= 61:

            print(x\_train)

            print(y\_train)

            print()

    # преобразуем в array

    x\_train, y\_train = np.array(x\_train), np.array(y\_train)

    # меняем размеры

    x\_train = np.reshape(x\_train, (x\_train.shape[0], x\_train.shape[1], 1))

    return x\_train, y\_train

def predict\_ta\_lstm(df):

    # загрузка данных

    #df = load\_data(symbol, interval)

    # подготовка данных

    data = df

    # добавим несколько технических индикаторов, таких как RSI и скользящие средние

    data['rsi'] = ta.rsi(data.Close, length=15)

    data['ema\_20'] = ta.ema(data.Close, length=20)

    data['ema\_50'] = ta.ema(data.Close, length=50)

    # drop NaN

    data = data.dropna(axis=0)

    print("data2",data.head)

    y = data['Close']

    del data['Close']

    del data['Dividends']

    del data['Stock Splits']

    print("data3",data.head)

    # scaling data

    x\_scaler = MinMaxScaler(feature\_range=(0,1)).fit(data)

    x\_scaled = x\_scaler.transform(data)

    y\_scaler = MinMaxScaler(feature\_range=(0,1)).fit(y.values.reshape(-1, 1))

    y\_scaled = y\_scaler.transform(y.values.reshape(-1, 1))

    # разделим на обучение и проверку

    test\_size = 0.3

    len\_x = len(data)

    x\_train = x\_scaled[:int(len\_x\*(1-test\_size))]

    y\_train = y\_scaled[:int(len\_x\*(1-test\_size))]

    x\_test = x\_scaled[int(len\_x\*(1-test\_size)):]

    y\_test = y\_scaled[int(len\_x\*(1-test\_size)):]

    # обучим модель

    model = Sequential()

    model.add(LSTM(10, input\_shape=(None, 1), activation='relu'))

    model.add(Dense(1))

    model.compile(optimizer='Adam', loss='mean\_squared\_error', metrics=[MeanAbsoluteError])

    history = model.fit(x\_train, y\_train, epochs=100)

    # предсказание на следуюший период

    data\_today = data.iloc[-1]

    data\_today\_scaled = x\_scaler.transform(data\_today.values.reshape(1, -1))

    predict\_today = model.predict(data\_today\_scaled)

    inversed = y\_scaler.inverse\_transform(predict\_today)

    return inversed