МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет вычислительной техники

Кафедра «Вычислительная техника»

Направление подготовки: 09.04.01 «Информатика и вычислительная техника»

Магистерская программа: «Прикладной искусственный интеллект»

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

на тему

ИСПОЛЬЗОВАНИЕ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ В ИНТЕЛЛЕКТУАЛЬНОМ АНАЛИЗЕ ДАННЫХ

Студент 6.06.24 Рудаков Павел Дмитриевич
(подпись, дата) (ФИО полностью)

Руководитель 6.06.24 Гудков П.А.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр 6.06.24 Бычков.А.С.
(подпись, дата) (фамилия, инициалы)

Рецензент Гл. специалист АО „НПП „Рубин" А.В. Васильков
(должность, место работы) (подпись, дата) (фамилия, инициалы)

*Работа допущена к защите (*протокол зас. кафедры от 27.05.24 № 11)

Заведующий кафедрой _____ М.А. Митрохин
(подпись) (фамилия, инициалы)

Работа защищена с отметкой отлично (протокол заседания ГЭК от 14.06.24 № 15)

Секретарь ГЭК _____ Е.И. Гурин
(подпись) (фамилия, инициалы)

Пенза 2024

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

Вычислительной техники

Кафедра

Вычислительная техника

«Утверждаю»
Заведующий кафедрой
_____М.А. Митрохин
«13» 09 2022 г.

ЗАДАНИЕ НА МАГИСТЕРСКУЮ ДИССЕРТАЦИЮ

1. Студент Рудаков Павел Дмитриевич, гр. 22ВВиМ1 факультета вычислительной техники, направления 09.04.01.

2.Тема работы: «Использование генетических алгоритмов в интеллектуальном анализе данных»

Тема утверждена приказом ПГУ № 765/о от «13» сентября 2022 г.

3. Руководитель магистерской диссертации к.т.н., доцент кафедры САПР Гудков П.А.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

Назначение разработки: Использование генетических алгоритмов для создания модели прогнозирования курса биткоина.

Моделирование и обучение нейросетевого алгоритма для решения задачи прогнозирования курса биткоина.

5. Перечень подлежащих разработке вопросов

– исследовать существующие архитектуры нейросетевых алгоритмов и современные инструменты для работы с нейронными сетями, отобрать наиболее подходящие.

– провести анализ исходных данных, на их основе сформировать обучающий набор.

– обучить подобранные модели нейронных сетей и провести

сравнительный анализ результатов обучения.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Исследование существующих архитектур нейросетевых алгоритмов и современных инструментов для работы с нейронными сетями	15%	13.09.2022 – 18.12.2022	
2.	Проведение анализа исходных данных и формирование обучающего набора	35%	19.12.2022 – 11.06.2023	
3.	Обучение подобранных моделей нейронных сетей, проведение сравнительного анализа результатов обучения	20%	12.06.2023 – 15.10.2023	
4.	Проектирование и разработка модулей системы	20%	16.10.2023 – 17.03.2024	
5.	Оформление пояснительной записки	10%	18.03.2024 – 19.05.2024	

Дата выдачи задания «13» сентября 2022 г.

Руководитель магистерской диссертации _____ Гудков П.А.
(подпись, дата)

Задание к исполнению принял студент _____ Рудаков П.Д.
(подпись, дата)

Работу к защите допустить
Декан факультета _____ 11.06.2024 _____ Фионова Л.Р.
(подпись, дата)

РЕФЕРАТ

Магистерская работа содержит 98 листов, 7 рисунков, 0 таблицы, 28 источников, 3 приложения

МАШИННОЕ ОБУЧЕНИЕ, ГЕНЕТИЧЕСКИЕ АЛГОРИТМЫ, МОДЕЛЬ LSTM, НЕЙРОННЫЕ СЕТИ, PYTHON, SCIKIT-LEARN

Цель работы - исследование алгоритмов машинного обучения с использованием генетических алгоритмов, выявление наиболее подходящей модели для решения задачи в интеллектуальном анализе данных. Объектом исследования является комплекс алгоритмов машинного обучения, основанных на нейронных сетях.

В результате проведенной работы были исследованы различные нейросетевые алгоритмы, выявлена наиболее успешная модель, обучен конкретный экземпляр сети, а также спроектирована интеллектуальная экспертная система для прогнозирования курса биткоина.

Для исследования и реализации было применено большое количество современных технологий для разработки нейронных сетей, такие как язык программирования Python, а также ряд библиотек: scikit-learn, numpy, matplotlib, Theano, Tensorflow, Deep.

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ 5
ВВЕДЕНИЕ 6
1. Введение в генетические алгоритмы и их применение в прогнозировании курса биткоина 9
1.1 Основные принципы генетических алгоритмов 9
1.2 Основные принципы генетических алгоритмов 13
1.3 Прогнозирование курса биткоина с помощью генетических алгоритмов 15
2. Анализ и предобработка исторических данных о курсе биткоина 18
2.1 Сбор исторических данных о курсе биткоина 18
2.2 Анализ временных рядов и поиск трендов в данных 20
2.3 Предобработка данных для использования в генетических алгоритмах 22
2.4 Биткоин и его оценка на крипторынке. 24
3. Прогнозирование рынка криптовалют с помощью алгоритмов искусственного интеллекта 32
3.1 Архитектура нейронных сетей для задач прогнозирования 32
3.2 Программирование нейронных сетей с помощью библиотек Python . 37
3.3 Основная функция генетических алгоритмов 51
3.3.1 Инициализация популяции 51
3.3.2 Оценка 52
3.3.3 Селекция 52
3.3.4 Скрещивание и мутация 52
3.3.5 Преобразование популяции и завершение 52
4. Выводы о преимуществах и недостатках генетического алгоритма в прогнозировании курса биткоина 53
ЗАКЛЮЧЕНИЕ 57
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 59
ПРИЛОЖЕНИЕ А Листинг программы. 65
ПРИЛОЖЕНИЕ Б UML-диаграммы 99
ПРИЛОЖЕНИЕ В Презентация 104

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

В настоящей магистерской диссертации применяются следующие сокращения и обозначения

БД – база данных

ОС – операционная система

ПО – программное обеспечение

СУБД – система управления базой данных

ЯП – язык программирования

API (Application programming interface) – программный интерфейс приложения

HTTP (HyperText Transfer Protocol) – протокол прикладного уровня передачи данных

JSON (JavaScript Object Notation) – формат обмена данными, который основан на языке программирования JavaScript

NoSQL (not only SQL) – это семейство нереляционных баз данных

REST (Representational State Transfer) – архитектурный стиль взаимодействия компонентов распределённого приложения в сети

ВВЕДЕНИЕ

В современном мире цифровая экономика стремительно набирает популярность. Сегодня практически каждая страна развивает это направление, хотя и с разной степенью успеха. Особенно сложно это дается развивающимся странам, так как внедрение элементов цифровой экономики требует значительных материальных вложений и обширной информационной базы.

В наше время владение информацией становится критически важным, так как каждую секунду происходят значимые события. Люди двадцать первого века ощущают потребность быть в курсе всех происходящих событий. Например, если пользователь не зайдет в социальные сети хотя бы на один день, его ждет огромное количество новостей на следующий день. Это подчеркивает, насколько быстро развивается наш мир – изменения происходят ежесекундно. Страны с недостаточно развитым Интернетом, к сожалению, отстают от тех, кто уже внедряет цифровую экономику на национальном уровне. Сегодня цифровизация затрагивает практически все аспекты человеческой деятельности. Взять, к примеру, образование: студенты в школах, колледжах и университетах активно используют Интернет для поиска информации, а онлайн-обучение стремительно развивается – многие университеты переходят на дистанционный формат лекций. Кроме того, цифровизация охватывает науку, медицину, строительство, рекламу и множество других сфер.

Для современных предприятий критически важно определять свою рыночную нишу, то есть тот социально-экономический слой людей, который наиболее заинтересован в их товарах или услугах. Однако эта задача непроста по ряду причин: отсутствие объективной информации о предпочтениях потребителей, сложность выделения вариативных показателей для группировки клиентов, а также нелинейные зависимости между показателями.

Таким образом, традиционные методы анализа и моделирования не всегда эффективны. Это приводит к сложности и неоднозначности маркетинговых решений, а также к скептицизму по отношению к строгим математическим методам, часто используемым экономистами для финансового анализа.[1] В то же

время, прогнозирование экономического эффекта от принятия тех или иных решений остается одной из самых важных задач.

Искусственный интеллект (ИИ, artificial intelligence) в определении Ассоциации развития искусственного интеллекта (Association for the Advancement of Artificial Intelligence, AAAI) – это «научное понимание механизмов, лежащих в основе мышления и интеллектуального поведения и их воплощение в машинах» [1]. Технологии ИИ, строятся на функционировании компьютерно-базирующихся искусственных нейронных сетей, обучающихся решению задач на основе анализа больших массивов данных. Это сети «нейронов» (вычислительных элементов, «процессоров»), преобразующих входные сигналы сети в выходные сигналы сети как решения некоторой задачи, –распознавания и генерирования объектов (образов, текстов, звука, событий и других). ИИ – значимый фактор функционирования экономики эпохи цифровой глобализации, обозначившейся, по данным компании глобального управленческого консультирования McKinsey, в 2014 г. и характеризующейся замещением трансграничных потоков товаров, финансов и услуг быстрорастущими цифровыми потоками [2]. McKinsey оценивает рост объёма потока данных в мире в период с 2005 по 2014 г. как сорока пятикратный. ИИ обеспечивают анализ, обработку и использование информации огромных и быстрорастущих информационных потоков в интересах частных лиц, групп, организаций, – в т.ч. бизнеса, корпораций, правительств, стран и других. Системы ИИ находят сегодня применение практически во всех отраслях экономики эпохи цифровой глобализации и в ближайшие годы уже все программные приложения будут содержать технологии ИИ.

По данным исследования компании Linkedin, образование, – вторая отрасль по востребованности навыков кандидатов в области разработки и использования систем ИИ после сферы программного обеспечения и ИТ услуг. Далее по степени убывания востребованности навыков ИИ следуют такие сферы как «аппаратные средства и сетевое оборудование», финансы, промышленное производство, потребительские товары, здравоохранение [3]. ИИ-навыки, – навыки, необходимые для создания ИИ-технологий, включая компетенции создания

нейросетей, глубокое обучение и машинное обучение, а также использование существующих систем ИИ.

Целью данной работы является рассмотрение инструментария, позволяющего повысить качество прогнозирования финансово- экономических показателей и, соответственно, уменьшить риски при продвижении товаров и услуг.

1. Введение в генетические алгоритмы и их применение в прогнозировании курса биткоина

1.1 Основные принципы генетических алгоритмов

Генетические алгоритмы являются один из наиболее мощных и эффективных инструментов в области оптимизации и прогнозирования. Они основаны на принципах естественного отбора и генетики, и представляют собой эволюционный подход к решению сложных задач. Генетические алгоритмы могут использоваться для решения широкого спектра задач, включая оптимизацию функций, поиск оптимальных параметров, обучение нейронных сетей и, конечно же, прогнозирование временных рядов.

Основные принципы генетических алгоритмов заключаются в имитации процессов естественного отбора и генетической мутации. Генетический алгоритм состоит из популяции индивидуальных решений, которые представляют собой наборы параметров или генов. Каждый индивидуум оценивается с помощью функции приспособленности, которая определяет его пригодность для решения задачи. Индивидуумы с более высокой приспособленностью имеют больший шанс выжить и передать свои гены следующему поколению [4].

Процесс эволюции в генетическом алгоритме состоит из нескольких шагов. Сначала происходит инициализация популяции начальными индивидуумами. Затем происходит оценка приспособленности каждого индивидуума. После этого выбираются родители для создания следующего поколения. Выбор родителей может быть случайным или основанным на приспособленности. Затем происходит скрещивание родителей, где происходит обмен генами и создание новых индивидуумов. Этот процесс повторяется до достижения заданного условия остановки, например, определенного числа поколений или достижения определенного значения приспособленности.

Генетические алгоритмы широко применяются в прогнозировании временных рядов, включая прогнозирование курса биткоина. Биткоин является самой популярной и широко используемой криптовалютой в мире, и его курс подвержен сильным колебаниям. Прогнозирование курса биткоина является

сложной задачей из-за множества факторов, влияющих на его изменение, таких как политическая ситуация, экономические показатели, новости и технические индикаторы.

Генетические алгоритмы могут быть использованы для создания моделей прогнозирования курса биткоина, которые могут учитывать множество факторов и адаптироваться к изменяющимся условиям. Входными данными для генетического алгоритма могут быть исторические данные о курсе биткоина, а выходными данными - прогнозируемые значения курса. Генетический алгоритм будет искать оптимальные параметры модели, которые наилучшим образом соответствуют историческим данным и могут быть использованы для прогнозирования будущих значений [5].

Одним из основных преимуществ генетических алгоритмов в прогнозировании курса биткоина является их способность к обнаружению сложных нелинейных зависимостей между входными и выходными данными. Курс биткоина может быть подвержен различным влияниям, которые могут быть сложно учесть с помощью традиционных методов прогнозирования. Генетические алгоритмы могут обнаруживать скрытые закономерности и нелинейные зависимости, которые могут быть использованы для более точного прогнозирования курса биткоина [6].

Генетические алгоритмы (ГА) имеют множество преимуществ перед другими методами подбора гиперпараметров, такими как сеточный поиск (Grid Search), случайный поиск, байесовская оптимизация и другие. Ниже описаны основные преимущества генетических алгоритмов;

исследование большого пространства поиска,

сеточный поиск ограничен дискретным набором параметров и становится неэффективным при увеличении их числа,

случайный поиск может охватывать пространство параметров неравномерно,

генетические алгоритмы, использующие популяции и эволюционные процессы, могут эффективно исследовать большие и сложные пространства параметров, повышая шансы на нахождение глобального оптимума,

избегание локальных минимумов,

В отличие от байесовской оптимизации, которая может застрять в локальных минимумах из-за своей модели, генетические алгоритмы могут пропустить локальные минимумы, используя процессы мутации и кроссовера для поиска глобального решения.

адаптивность и гибкость.

Генетические алгоритмы не требуют предположений о природе оптимизируемой функции и могут применяться к широкому кругу задач. Это делает их более универсальными, чем методы, требующие знания функции потерь или свойств данных.

Параллелизм

Генетические алгоритмы можно легко распараллелить, поскольку пригодность разных особей в популяции может оцениваться независимо. Это значительно ускоряет процесс поиска оптимальных гиперпараметров по сравнению с методами, требующими последовательной оценки.

Реализация и адаптация к различным задачам

Генетические алгоритмы могут быть адаптированы для решения широкого круга задач, включая непрерывные, дискретные и даже задачи со смешанными параметрами. Это делает их более гибкими для использования в различных ситуациях.

Решение сложных фитнес-функций

Генетические алгоритмы хорошо работают с фитнес-функциями, которые могут быть негладкими, многомерными или даже аналитически выразимыми. Это свойство особенно важно в реальных задачах машинного обучения, где фитнес-функция может быть сложной и неопределенной.

Сравнивая с другими методами, можно сделать выводы:

Сеточный поиск: простой, но неэффективный для больших размерностей.

Случайный поиск: лучше, чем поиск по сетке, но может не использовать хорошие наборы параметров.

Байесовская оптимизация: эффективна, но может застревать в локальных минимумах и требует большего количества времени на моделирование.

Эволюционные стратегии и другие эволюционные методы: генетические алгоритмы — это более широкий набор эволюционных методов, которые также обладают многими из вышеперечисленных преимуществ.

В заключение, генетические алгоритмы представляют собой мощный инструмент в области прогнозирования курса биткоина. Они основаны на принципах естественного отбора и генетики, и позволяют создавать модели для подбора гиперпараметров в модели прогнозирования, которые могут учитывать множество факторов и адаптироваться к изменяющимся условиям. Генетические алгоритмы способны обнаруживать сложные нелинейные зависимости и скрытые закономерности, что делает их идеальным инструментом для подбора гиперпараметров прогнозирования курса биткоина. В дальнейшем исследовании будет рассмотрено применение генетических алгоритмов для подбора гиперпараметров в прогнозировании курса биткоина на языке программирования Python.

1.2 Основные принципы генетических алгоритмов

В современном мире финансовые рынки играют огромную роль в экономике, и прогнозирование их движения является важной задачей для многих инвесторов и трейдеров. В последние годы биткоин стал одним из самых популярных активов на финансовых рынках, и его курс стал объектом пристального внимания многих инвесторов. Прогнозирование курса биткоина является сложной задачей, так как он подвержен значительным колебаниям и волатильности. В связи с этим, появилась потребность в разработке новых и эффективных методов прогнозирования курса биткоина.

Одним из таких методов являются генетические алгоритмы, которые позволяют моделировать и эмулировать процессы естественного отбора и эволюции в биологических системах. Генетические алгоритмы основаны на принципах генетики и эволюции, и используются для решения оптимизационных задач. Они представляют собой эффективный и гибкий инструмент, который может быть применен в различных областях, включая финансовую аналитику [13].

Применение генетических алгоритмов в финансовой аналитике позволяет решать такие задачи, как оптимизация портфеля, прогнозирование курсов валют и акций, анализ рынка и многое другое. В контексте прогнозирования курса биткоина, генетические алгоритмы могут быть использованы для создания моделей, которые способны предсказывать будущие значения курса на основе исторических данных [5].

Одной из основных проблем при прогнозировании курса биткоина является его высокая волатильность. Курс биткоина может сильно колебаться в течение короткого времени, что делает его прогнозирование сложным. Генетические алгоритмы позволяют учитывать эту волатильность и адаптироваться к изменяющимся условиям рынка. Они способны находить оптимальные параметры модели, которые могут давать наиболее точные прогнозы курса биткоина [6].

Основными компонентами генетического алгоритма являются популяция, генотипы и фенотипы. Популяция представляет собой набор индивидов, которые представляют возможные решения задачи. Генотипы являются внутренним

представлением индивидов, а фенотипы - внешним представлением. Генотипы кодируются в виде битовых строк, которые представляют значения параметров модели. Фенотипы представляют собой значения параметров, полученные из генотипов.

Генетический алгоритм состоит из нескольких этапов. На первом этапе создается начальная популяция, которая состоит из случайно сгенерированных индивидов. Затем происходит оценка каждого индивида с помощью функции приспособленности, которая определяет, насколько хорошо данный индивид приспособлен к решению задачи. Индивиды, которые имеют более высокие значения функции приспособленности, имеют больше шансов быть выбранными для следующего поколения.

Далее происходит операция скрещивания, в результате которой происходит обмен генетическим материалом между выбранными индивидами. Это позволяет создавать новых потомков, которые могут иметь комбинацию хороших качеств родителей. Затем происходит операция мутации, в результате которой происходит случайное изменение генетического материала у некоторых индивидов. Это позволяет вносить разнообразие в популяцию и избегать преждевременной сходимости к определенному решению.

После выполнения операций скрещивания и мутации происходит формирование новой популяции, которая замещает предыдущую. Процесс формирования новой популяции повторяется несколько раз, пока не будет достигнуто условие остановки, например, определенное количество итераций или достижение определенного значения функции приспособленности.

Применение генетических алгоритмов в прогнозировании курса биткоина имеет ряд преимуществ. Во-первых, генетические алгоритмы позволяют учитывать сложность и нелинейность рыночных данных, что является важным при прогнозировании курса биткоина. Во-вторых, они могут быть применены для оптимизации параметров модели, что позволяет достичь наилучших результатов прогнозирования. В-третьих, генетические алгоритмы являются гибким

инструментом, который может быть адаптирован к различным условиям рынка и изменяющимся трендам.

Однако, применение генетических алгоритмов в прогнозировании курса биткоина также имеет свои ограничения. Во-первых, генетические алгоритмы требуют большого количества вычислительных ресурсов и времени для выполнения. Во-вторых, прогнозирование курса биткоина является сложной задачей, и генетические алгоритмы могут не всегда давать точные и надежные результаты. В-третьих, генетические алгоритмы требуют определенного уровня экспертизы и опыта для их правильной настройки и интерпретации результатов.

В заключение, генетические алгоритмы представляют собой эффективный инструмент для прогнозирования курса биткоина. Они позволяют учитывать сложность и волатильность рыночных данных, оптимизировать параметры модели и адаптироваться к изменяющимся условиям рынка. Однако, их применение требует достаточного количества вычислительных ресурсов и времени, а также экспертизы и опыта для их настройки и интерпретации результатов. В дальнейшем исследовании можно рассмотреть различные варианты генетических алгоритмов и их комбинацию с другими методами прогнозирования для достижения более точных результатов.

1.3 Прогнозирование курса биткоина с помощью генетических алгоритмов

Генетические алгоритмы – это эффективный метод оптимизации, основанный на биологической эволюции. Они используются для решения сложных задач, включая прогнозирование курса биткоина. В данной работе мы исследуем применение генетических алгоритмов в прогнозировании курса биткоина с использованием языка программирования Python.

Прогнозирование курса биткоина является сложной задачей, так как он подвержен значительным колебаниям и зависит от множества факторов, включая политическую и экономическую ситуацию, технические инновации и изменения в правовом регулировании. Традиционные методы прогнозирования, такие как

статистические модели или анализ временных рядов, могут быть недостаточно эффективными при такой высокой степени сложности и неопределенности [17].

Генетические алгоритмы предлагают альтернативный подход к прогнозированию курса биткоина. Они основаны на принципах биологической эволюции и имитируют процесс отбора и размножения в популяции. Главная идея заключается в создании и эволюции популяции индивидуальных решений, называемых хромосомами, которые представляют собой наборы параметров или правил прогнозирования. Хромосомы кодируются в виде строк символов или чисел, и процесс эволюции осуществляется путем применения генетических операторов, таких как скрещивание и мутация, к родительским хромосомам [21].

Применение генетических алгоритмов в прогнозировании курса биткоина имеет несколько преимуществ. Во-первых, они способны работать с большим количеством параметров и учитывать их взаимодействие. Это позволяет моделировать сложные зависимости между различными факторами, которые могут влиять на курс биткоина. Во-вторых, генетические алгоритмы могут автоматически оптимизировать параметры прогнозной модели, находя наилучшие комбинации, которые обеспечивают наиболее точные прогнозы. Это особенно полезно в случае курса биткоина, где эффективность различных моделей может сильно варьироваться во времени. В-третьих, генетические алгоритмы могут учитывать неопределенность и случайность в данных, что делает их устойчивыми к шуму и выбросам [7].

Для реализации прогнозирования курса биткоина с использованием генетических алгоритмов на языке программирования Python мы будем использовать библиотеку DEAP (Distributed Evolutionary Algorithms in Python). DEAP предоставляет широкий набор инструментов для создания и эволюции генетических алгоритмов, включая генерацию начальной популяции, реализацию генетических операторов и оценку качества хромосом.

В процессе разработки прогнозной модели мы будем использовать исторические данные о курсе биткоина, а также различные факторы, которые могут влиять на его изменение. Например, мы можем включить в модель данные

о торговых объемах, новостях и событиях, связанных с биткоином, а также технические показатели, такие как скользящие средние или индикаторы относительной силы. Важно отметить, что выбор факторов и их веса в модели могут сильно варьироваться в зависимости от времени и условий рынка, поэтому генетический алгоритм позволяет автоматически оптимизировать эти параметры.

В заключение, прогнозирование курса биткоина с помощью генетических алгоритмов представляет собой интересную и перспективную область исследования. Она позволяет учитывать сложность и неопределенность в данных, а также оптимизировать параметры модели для достижения наилучших результатов. Разработка и реализация прогнозной модели на языке программирования Python с использованием библиотеки DEAP позволит нам более глубоко изучить эту тему и оценить эффективность генетических алгоритмов в прогнозировании курса биткоина.

2. Анализ и предобработка исторических данных о курсе биткоина

2.1 Сбор исторических данных о курсе биткоина

Сбор исторических данных о курсе биткоина является важным этапом в разработке модели прогнозирования. Для этого можно использовать различные источники данных, такие как биржи криптовалют, финансовые платформы и специализированные сервисы. Одним из наиболее популярных источников данных является API (Application Programming Interface), предоставляемый различными биржами.

API позволяет получать доступ к историческим данным о курсе биткоина и других криптовалют, а также к другой полезной информации, такой как объемы торгов, статистика и т.д. Для использования API необходимо зарегистрироваться на соответствующей бирже и получить API-ключ, который будет использоваться для аутентификации при каждом запросе [15].

После получения API-ключа можно приступить к сбору данных. Для этого необходимо написать скрипт на языке Python, который будет отправлять запросы к API и сохранять полученные данные. Например, можно отправить запрос на получение исторических данных о курсе биткоина за определенный период времени с определенным интервалом (например, каждый час или каждый день). Полученные данные можно сохранить в формате CSV или JSON для последующего анализа и обработки [8].

При сборе данных необходимо учитывать несколько важных аспектов. Во-первых, необходимо определить период, за который будут собираться данные. Этот период может быть различным в зависимости от целей исследования. Например, можно собирать данные за последний год или за все доступные исторические данные [8].

Во-вторых, необходимо определить интервал сбора данных. Этот интервал может быть различным в зависимости от того, какую детализацию данных требуется. Например, можно собирать данные каждый час, каждый день или

каждую неделю. Важно учитывать, что более детализированные данные могут занимать больше места и требовать больше времени на сбор.

После сбора данных необходимо провести их предобработку. Предобработка данных включает в себя несколько этапов. Во-первых, необходимо проверить данные на наличие пропущенных значений. Пропущенные значения могут быть вызваны различными причинами, такими как ошибки в данных или проблемы с источником данных. Важно обработать пропущенные значения, чтобы они не искажали результаты анализа.

Во-вторых, необходимо провести анализ выбросов. Выбросы могут быть вызваны аномальными значениями в данных или ошибками в сборе данных. Выбросы могут искажать результаты анализа и прогнозирования, поэтому важно их обнаружить и обработать. Для обнаружения выбросов можно использовать различные статистические методы, такие как метод межквартильного расстояния или метод z-оценки.

В-третьих, необходимо провести анализ временных рядов. Временной ряд представляет собой последовательность значений, упорядоченных по времени. Анализ временных рядов позволяет выявить закономерности и тренды в данных, что может быть полезно для прогнозирования. Для анализа временных рядов можно использовать различные методы, такие как скользящее среднее, экспоненциальное сглаживание или авторегрессионные модели.

В-четвертых, необходимо провести статистический анализ данных. Статистический анализ позволяет выявить связи и зависимости между различными переменными. Например, можно провести корреляционный анализ, чтобы выявить связь между курсом биткоина и другими факторами, такими как объемы торгов или новостные события. Это может помочь в построении более точной модели прогнозирования.

В-пятых, необходимо провести визуализацию данных. Визуализация данных позволяет наглядно представить результаты анализа и обнаружить закономерности, тренды и выбросы. Для визуализации данных можно

использовать различные графические инструменты, такие как графики, диаграммы, гистограммы и др.

В итоге, анализ и предобработка исторических данных о курсе биткоина является важным этапом в разработке модели прогнозирования. Сбор данных позволяет получить информацию о курсе биткоина за определенный период времени, а предобработка данных позволяет обработать их, чтобы они были готовы для анализа и построения модели прогнозирования. Анализ данных позволяет выявить закономерности и тренды, а визуализация данных позволяет наглядно представить результаты анализа. Все эти этапы являются важными для построения точной и эффективной модели прогнозирования курса биткоина с помощью генетических алгоритмов на языке Python.

2.2 Анализ временных рядов и поиск трендов в данных

Анализ и предобработка исторических данных о курсе биткоина являются важным этапом в прогнозировании его будущего значения. В данном разделе мы рассмотрим основные методы анализа временных рядов и поиска трендов в данных, которые помогут нам лучше понять историческую динамику курса биткоина.

Первым шагом в анализе и предобработке данных является загрузка исходных данных о курсе биткоина. Для этого мы можем воспользоваться различными источниками, такими как финансовые платформы, биржи или специализированные сервисы, предоставляющие исторические данные о курсе биткоина. Важно выбрать достоверный и надежный источник данных, чтобы получить точные и актуальные значения курса [9].

После загрузки данных мы можем приступить к их анализу. Одним из первых шагов является визуализация временного ряда курса биткоина. Для этого мы можем использовать графические инструменты, такие как графики и диаграммы, чтобы наглядно представить динамику курса. Визуализация помогает нам увидеть основные тренды и паттерны в данных, а также выявить аномалии или выбросы [4].

Для анализа временных рядов мы можем использовать различные методы и техники, такие как статистический анализ, автокорреляционная функция (ACF), сезонность и т.д. Статистический анализ позволяет нам оценить основные характеристики данных, такие как среднее значение, стандартное отклонение и медиана. Автокорреляционная функция (ACF) позволяет нам определить наличие автокорреляции в данных, то есть зависимости между значениями в разные моменты времени.

Сезонность является важным аспектом в анализе временных рядов. Она позволяет нам выявить периодические колебания в данных, которые могут быть связаны с определенными сезонными факторами или циклами. Для выявления сезонности мы можем использовать методы декомпозиции временного ряда, такие как аддитивная или мультипликативная модели. Декомпозиция позволяет разделить временной ряд на трендовую, сезонную и остаточную компоненты, что помогает нам лучше понять динамику данных.

Поиск трендов в данных является еще одним важным аспектом анализа временных рядов. Тренд представляет собой долгосрочное направление движения курса биткоина и может быть восходящим, нисходящим или боковым. Для поиска трендов мы можем использовать различные методы, такие как скользящее среднее, экспоненциальное сглаживание и регрессионный анализ. Скользящее среднее позволяет нам сгладить временной ряд, чтобы выявить долгосрочные тренды, в то время как экспоненциальное сглаживание учитывает веса более новых значений. Регрессионный анализ позволяет нам оценить зависимость между курсом биткоина и другими факторами, такими как объем торгов или новостные события.

Важно отметить, что анализ и предобработка исторических данных о курсе биткоина являются лишь первым шагом в прогнозировании его будущего значения. Для более точного прогноза мы можем использовать различные методы машинного обучения, такие как генетические алгоритмы, нейронные сети или ансамбли моделей. Эти методы позволяют нам учесть сложные зависимости и

взаимодействия между различными факторами, что помогает нам создать более точные и надежные прогнозы курса биткоина.

В заключение, анализ и предобработка исторических данных о курсе биткоина являются важным этапом в прогнозировании его будущего значения. Они позволяют нам лучше понять динамику и тренды в данных, а также выявить основные факторы, влияющие на курс биткоина. Использование различных методов анализа временных рядов и поиска трендов позволяет нам создать более точные и надежные модели прогнозирования курса биткоина.

2.3 Предобработка данных для использования в генетических алгоритмах

Анализ и предобработка исторических данных о курсе биткоина являются важными шагами в разработке модели прогнозирования курса биткоина с использованием генетических алгоритмов. В данном разделе мы рассмотрим основные методы анализа и предобработки данных, а также подробно остановимся на специфических требованиях предобработки для использования в генетических алгоритмах.

Первым шагом в анализе данных о курсе биткоина является сбор исторических данных. Исторические данные представляют собой информацию о цене биткоина в разные моменты времени. Для этого можно использовать различные источники данных, такие как биржи, специализированные платформы и финансовые ресурсы. Важно убедиться, что данные достоверны и имеют высокую степень точности [10].

После сбора данных необходимо провести их предварительный анализ. В рамках анализа данных мы можем оценить различные статистические характеристики, такие как среднее значение, медиана, стандартное отклонение и корреляция. Эти характеристики помогут нам понять основные тенденции и закономерности в данных.

Для предобработки данных перед использованием в генетических алгоритмах необходимо выполнить несколько шагов. Первым шагом является удаление выбросов. Выбросы могут искажать результаты и приводить к

неправильным выводам. Для выявления выбросов можно использовать различные статистические методы, такие как правило трех сигм.

Далее необходимо заполнить пропущенные значения. Пропущенные значения могут возникать по разным причинам, например, из-за ошибок в сборе данных или отсутствия информации в определенные периоды времени. Заполнение пропущенных значений можно выполнить различными способами, например, средним значением или интерполяцией.

После заполнения пропущенных значений необходимо провести нормализацию данных. Нормализация данных поможет привести их к одному масштабу и избежать проблем с весами при использовании генетических алгоритмов. Одним из распространенных методов нормализации является минимаксное масштабирование, при котором значения данных приводятся в интервал от 0 до 1.

После нормализации данных можно приступить к выбору признаков. Выбор признаков является важным этапом, так как он определяет, какие данные будут использоваться для обучения модели. Некоторые признаки могут быть неинформативными или коррелировать с другими признаками, поэтому их следует исключить. Для выбора признаков можно использовать различные методы, такие как анализ главных компонент или отбор признаков на основе статистических тестов.

После выбора признаков данные готовы для использования в генетических алгоритмах. Генетические алгоритмы являются эффективным инструментом для прогнозирования курса биткоина, так как они могут учитывать сложные взаимосвязи между различными признаками и находить оптимальные комбинации для предсказания курса.

В заключение, анализ и предобработка исторических данных о курсе биткоина являются важными этапами в разработке модели прогнозирования курса биткоина с использованием генетических алгоритмов. Правильная предобработка данных, включающая удаление выбросов, заполнение пропущенных значений, нормализацию и выбор признаков, позволит получить надежные и точные

результаты прогнозирования. Генетические алгоритмы, в свою очередь, обеспечат эффективное использование данных и поиск оптимальных комбинаций для предсказания курса биткоина.

2.4 Биткоин и его оценка на крипторынке

В последнем разделе второй главы мы рассмотрим основные аспекты криптовалют, их характеристики, правовое положение в России и разнообразие видов. Особое внимание уделяется отсутствию нормативной правовой базы, регулирующей взаимодействие между пользователями, что подчеркивает необходимость не только детального научного изучения криптовалют, но и разработки институциональных норм. В статье обсуждается положение криптовалют в мировой экономике, приводится информация о динамике, обороте и прогнозах. На основе проанализированного материала я сделал выводы об общих положениях криптовалют на мировом рынке.

XXI век – эпоха стремительного развития и технического прогресса. С каждым днем инновации все больше входят в нашу жизнь. Одной из ключевых ролей в современной жизни играют деньги, и было бы удивительно, если бы они не претерпели изменений. Одной из новейших и еще не до конца изученных инноваций в «мире экономики» является криптовалюта. Это понятие появилось в жизни онлайн-пользователей около восьми лет назад, но его значение и применение известны лишь немногим. Общественное понятие криптовалюты подразумевает цифровые денежные средства, которые не только хранятся в электронных кошельках, но и переводятся между ними. Проще говоря, криптовалюта – это разновидность цифровой валюты. Термин стал известен после публикации статьи «Cryptocurrency» о системе Биткоин в 2011 году в журнале Forbes. Автор, как и многие другие, использовал термин «электронная наличность». Для эмиссии разные криптовалюты применяют различные методы: майнинг, форжинг или ICO. В целом, криптовалюта практически не отличается от обычных компьютерных кодов, через которые происходит шифрование денежных переводов. Создать и использовать валюту может любой пользователь. Разница лишь в том, что криптовалюта, в отличие от стандартных денежных средств, не

просто запускается в сети, а фактически появляется в ней. Криптознаки создаются без вмешательства какого-либо органа, следовательно, криптовалюта не имеет эмиссионного центра (отсутствует банк, который выпускает денежные знаки). Криптознаки генерируются вычислительной техникой, включая персональные компьютеры. Все криптознаки находятся на кошельках реальных пользователей.

Криптовалюты не накапливаются в общем хранилище, они распределены по кошелькам пользователей. Комиссия за переводы отсутствует, что устраняет необходимость в сотрудничестве. Исключением являются биржи, через которые криптовалюта может быть переведена в реальные средства. Информация о транзакциях доступна в открытом виде и не зашифрована. Учет криптовалюты децентрализован, работа систем основана на технологии блокчейна – непрерывной последовательной цепочке блоков, содержащих информацию. Ни один пользователь не может оспорить, отменить или заблокировать передачу криптовалюты. Участники сделки могут добровольно заблокировать свои криптовалюты или установить условие, что для завершения или отмены сделки необходимо согласие определенных участников.

С экономической точки зрения, основная особенность криптовалюты – это деньги, не подверженные инфляции, так как их количество фиксировано. Курс валюты напрямую зависит от спроса: чем больше покупателей, тем выше цена, и наоборот. Никто не может регулировать курс криптовалюты, управлять ею или изменять количество. Эти свойства делают криптовалюту независимой от экономических и политических ситуаций в стране.

История криптовалюты для секретных платежей началась с использования системы DigiCash Дэвида Чома в 1990 году. Компания быстро обанкротилась, и система прекратила существование. От современной криптовалюты система отличалась централизованностью. В 2008 году обсуждения криптовалют начались в узком кругу IT-специалистов. Стартом послужила статья о первой виртуальной денежной единице, основанной на криптографии – Биткоине. Сатоши Накамото предложил свою версию криптоденег, и вскоре была запущена сеть Bitcoin вместе с первыми кошельками. Начался процесс майнинга биткоина. Многие IT-

специалисты не понимали значения и перспектив криптовалюты: для одних это была забава, для других – деньги будущего. Вероятно, никто, кроме создателя криптовалюты, не осознавал, каких высот она достигнет.

Сатоши Накамото майнил биткоины на особом оборудовании, что помогло завладеть ему 1 миллионом монет. Власти РФ переносят возможность легализации операций с криптовалютами в 2018 году в рамках борьбы с незаконными денежными переводами. [2] Государство должно знать каждого участника финансовой операции. Если это сделка, то лицам, принимающим в ней непосредственное участие, необходимо понимать, у кого они покупают и кому они продают, как это происходит при банковских операциях. Алексей Моисеев [2]. Банк России предложил ввести налогообложение криптовалюты и идентифицировать ее в качестве цифрового товара. Данное предложение поддержал Министр Экономического развития, Министр финансов и «силовые ведомства». В июне 2017 года зампред Центробанка Ольга Скоробогатова сообщила о планах Центробанка разработать в России собственную виртуальную валюту. Заместитель представителя Центробанка Ольга Скоробогатова поставила в известность о планах Центробанка разработать собственную криптовалюту в России. «До виртуальной национальной валюты мы точно дойдем. Над этим мы уже начали работать. На самом деле все регуляторы пришли к мысли, что, безусловно, национальную виртуальную валюту делать надо. Это будущее. Вопрос конкретного времени.» Ольга Скоробогатова [2] 10 октября 2017 года на совещании президента РФ было принято решение о регулировании государством процессов майнинга, эмиссии о обращения криптовалют. [2] Об юридическом статусе и сути валюты ведутся рассуждения во многих странах. Они рассматривают криптовалюту по– разному: особый товар, средство оплаты. Также могут присутствовать ограничения в обороте. Обговорив значение криптовалюты, историю ее создания, особенности и правовое значение в РФ, мы можем рассмотреть наиболее популярные виды криптовалюты на данное время.

Bitcoin (BTC) – первая и наиболее известная криптовалюта. Рыночная капитализация биткоина составляет 40 миллиардов долларов, и он является самой

дорогой валютой на рынке. Bitcoin признан на государственном уровне в ряде стран, имеет оффлайн-терминалы и используется в качестве инвестиционного инструмента. Ethereum (ETH) – электронная монета, созданная Виталием Бутериным. Эфириум (эфир) использует собственный исходный код, не базируясь на биткоине. Курс эфира достиг $222 за 1 ETH, а капитализация – 20 миллиардов долларов, что делает его серьезным конкурентом биткоину. Ripple (XRP) – криптовалюта, появившаяся в 2012 году. По капитализации она занимает третье место с 11,5 миллиарда долларов, а курс составляет $0,29 за монету. Протокол Ripple часто используется банками и платежными системами благодаря своим преимуществам перед биткоином, таким как моментальные переводы и отсутствие комиссии. Litecoin (LTC) – форк биткоина, запущенный в 2011 году, который уже приобрел значительную популярность. Курс Litecoin составляет $27, а капитализация – $1,4 миллиарда. NEM (XEM) – криптовалюта второго поколения, позиционирующая себя как "социально-ориентированная". При курсе $0,22 рыночная капитализация NEM превышает 2 миллиарда долларов.

Из-за отсутствия контроля над криптовалютами они активно используются в личных целях преступными элементами, что вызывает скептицизм у многих мировых держав. В то же время в странах Евросоюза криптовалюты имеют статус договорного средства платежа, а в Японии они признаны официально и интегрируются в банковскую систему. В Китае заметна тенденция к широкому использованию криптовалют, предпочтение часто отдают биткоинам вместо американского доллара. В странах третьего мира криптовалюты запрещены, но их присутствие в сети продолжается. Несмотря на историческую нестабильность валют, криптовалюта демонстрирует значительный рост и привлекает все большее внимание. Покупка криптовалют становится способом защиты от девальвации национальной валюты. Мировой оборот криптовалюты составляет порядка $4 миллиардов, причем основной оборот биткоинов происходит на валютных биржах, где их покупают и продают за евро, рубли и доллары. Существует более пятидесяти активных bitcoin-бирж. Многие интернет-службы и онлайн-магазины принимают оплату за товары и услуги в криптовалюте.

Несмотря на оптимистичные прогнозы, недавний запрет Центробанка показывает, что современная экономика пока не готова принять децентрализованную валюту. Криптовалюта – современное и интересное решение многих проблем, таких как независимость от экономической и политической ситуаций в стране. Однако многие государства пока не принимают её, и она успешно реализуется лишь в странах с высокоразвитой экономикой. Вероятно, в будущем многие страны пересмотрят своё отношение к криптовалютам и они выйдут на мировой рынок.

Бумажные деньги и криптовалюты предназначены для обмена товарами и услугами, и у каждой из них есть свои плюсы и минусы. Современные технологии улучшают и упрощают нашу жизнь, и наряду с бумажными и электронными деньгами появляются криптовалюты. Самой известной и быстрорастущей, но спорной, является биткоин. Обычные люди мало знают о биткоине, кроме того, что он дорогой, не имеет физической формы и связан с видеокартами. Биткоин уже оказывает влияние на экономику и может стать основной валютой для сделок центральных и коммерческих банков, у которых будут собственные криптовалюты.

Биткоин (BTC) можно обменять на доллары, евро, рубли и другие валюты, а также использовать для покупок в интернете или переводов. Однако у него нет физической формы, и как у любой валюты, у биткоина есть свои преимущества и недостатки. К преимуществам относятся анонимность, низкие комиссии и высокая скорость переводов. Комиссия за перевод bitcoin составляет 0,0001 BTC, что эквивалентно примерно 2 рублям. Время перевода может составлять всего несколько минут. Отсутствие возможности отмены транзакции может быть как достоинством, так и недостатком. Главным достоинством биткоина является невозможность его подделки. Однако к недостаткам можно отнести отсутствие стабильности курса и то, что не все страны готовы принять криптовалюту. Это делает биткоин подходящим для спекуляций, но не для долгосрочных вложений.

Ситуация с криптовалютами продолжает развиваться, и в будущем они могут занять значимое место на мировом финансовом рынке.

Подводя итог всех преимуществ и недостатков данной криптовалюты, можно сказать, что биткоин подходит для тех, кто хочет анонимно и быстро переводить крупные суммы денег с минимальной комиссией, но для хранения сбережений эта криптовалюта не подходит. У биткоина нет эмитента – юридического лица, которое его выпускает, в отличие от, например, рубля, эмитируемого Центробанком. Центробанк сам определяет, сколько денег нужно напечатать. У биткоина такого нет. Также нет единого места хранения данных об этой криптовалюте. Эти данные хранятся на множестве компьютеров одновременно, подобно торрент-файлам. Таким образом, биткоины существуют не как физические объекты, а как информация о транзакциях.

Использование банковской карты сопряжено с множеством комиссий и рисками ареста счетов судебными приставами или их заморозки банком. В случае биткоина таких рисков нет, так как он основан на блокчейне, а не на банке. Блокчейн (цепочка блоков) – это упорядоченная база данных, где данные хранятся не на общем сервере, а распределенно, в виде постоянно растущего списка записей, называемых блоками. Каждый блок содержит метку времени и ссылку на предыдущий блок. Шифрование обеспечивает, что только владельцы закрытых ключей могут изменять свои записи, а также синхронизацию копий блокчейна у всех пользователей. Безопасность заложена в самой архитектуре блокчейна. Концепцию блокчейна предложил Сатоши Накамото в 2008 году, и впервые она была реализована в 2009 году как часть биткоина, где блокчейн служит главным реестром всех транзакций.

Благодаря технологии блокчейна биткоин стал первой цифровой валютой, решающей проблему двойных расходов без использования центрального органа. Блокчейн также помогает идентифицировать пользователей и поддерживать доверие между ними, так как никто не может изменить данные в блоке без нужных ключей. Сегодня для передачи информации все пользуются интернетом, но для передачи ценностей, таких как деньги, приходится обращаться к устаревшим финансовым учреждениям, таким как банки. Даже современные способы онлайн-оплаты требуют соединения с банковским счетом или кредитной картой.

Технология блокчейн позволяет устранить этих посредников, ведя учет транзакций и устанавливая личность сторон без участия банков.

Криптовалюта биткоин, несмотря на отсутствие физической формы, одновременно соответствует и не соответствует традиционным представлениям о деньгах. Возникает вопрос, почему такой актив, представленный лишь зашифрованным кодом, может быть столь высоко оценен, ведь он не обеспечен другими валютами или материальными ценностями и основан только на взаимном доверии участников системы. Биткоин можно рассматривать как попытку создания частных денег, независимых от централизованных аналогов. Биткоин – это не только валюта, но и система денежных переводов. Участники системы, генерируя блоки, зарабатывают биткоины, за что им начисляется небольшая комиссия за включение трансфера в блокчейн. Традиционные финансовые институты с их высокими операционными расходами не могут конкурировать с биткоином.

Вычислительные узлы, в отличие от банков, заинтересованы в развитии системы, и даже после завершения генерации виртуальных монет они будут получать доход. К 2150 году, когда будет выпущена последняя монета биткоина, он станет всемирным и дорогим активом с суточным оборотом в десятки миллиардов долларов. Биткоин является дефляционным активом, но это проявляется в долгосрочной перспективе. После завершения генерации виртуальных монет биткоин станет более стабильным. Его ограниченная эмиссия и предсказуемость обеспечивают стабильность, однако высокая волатильность пока остается проблемой. Цена на биткоин зависит от спроса и предложения, и его значительная часть потеряна навсегда из-за утраты доступа к кошелькам.

В будущем биткоин может решить проблему инфляционного налога, но вызовет дефляционный аналог, когда цифровая валюта будет перемещаться от продавцов к инвесторам. Инфляция стимулирует экономический рост, в то время как дефляция может тормозить его. В бедных странах, страдающих от девальвации национальных валют, биткоин служит средством сохранения и увеличения сбережений. Однако для стран с развитой экономикой аргумент о

неограниченном печатании денег выглядит несуразно, особенно на фоне политики количественного смягчения и отрицательных процентных ставок.

В странах с развитой экономикой и устойчивой финансовой системой криптовалюты займут свою нишу, но их использование будет контролироваться финансовыми регуляторами. В развивающихся странах необходим жесткий контроль при их внедрении. Bitcoin в своей нынешней форме не может стать заменой фиатных денег и основной валютой мировой экономики. Даже если регуляторы примут bitcoin, дефляция и другие факторы будут препятствовать развитию криптовалютной экономики. На данный момент биткоином чаще пользуются для заработка традиционных денег, чем для обычного обмена.

Чтобы создать экономику на основе криптовалют, необходимо внести изменения, которые сохранят их преимущества и сделают их более похожими на традиционные деньги.

3. Прогнозирование рынка криптовалют с помощью алгоритмов искусственного интеллекта

3.1 Архитектура нейронных сетей для задач прогнозирования

Искусственный интеллект (ИИ) издавна использовался для прогнозирования различных событий. Алгоритмы, основанные на исторических данных, успешно предусматривают победителей Оскара, цены на акции и даже распространение пандемии.

В течение последних пяти лет предпринимались попытки спрогнозировать курс биткойны к доллару. Стартапы, исследовательские группы и энтузиасты все еще пытаются создать алгоритм, который мог бы предсказать поведение цифрового золота на фондовом рынке в краткосрочной и долгосрочной перспективе.

Разработчики алгоритмов машинного обучения используют различные подходы для создания инструментов прогнозирования. Самыми популярными из них являются повторяющаяся нейронная сеть и модель долгосрочной кратковременной памяти (LSTM).

LSTM – это тип повторяющихся нейронных сетей, которые могут запоминать долгосрочные зависимости. Подобно тому, как мы используем предыдущий опыт для прогнозирования будущих событий, нейронная сеть способна долго запоминать информацию и быстро находить закономерности.

Первые попытки прогнозировать цену биткоинов были сделаны во время бума, который состоялся в конце 2017 года. Сербский предприниматель и разработчик Огнен Гатала создал бота в Twitter, который должен был размещать расчетные котировки биткоинов течение следующих нескольких дней, используя исторические данные. Для этого он использует API blockchain. info (ныне blockchain. com), где алгоритм собирает данные о ценах за последние два месяца и пытается «угадать» будущие колебания, используя метод поиска ближайших соседей.

Однако Огнен Гатала допустил достаточно серьёзную ошибку. Алгоритм учитывал только исторические данные о ценах, тогда как на цену влияет много других факторов.

Еще одну попытку сделал в 2019 году американский разработчик алгоритмов машинного обучения Марко Сантос. Марко использовал LSTM как инструмент прогнозирования, а источником данных избрал Yahoo Finance. По замыслу разработчика, алгоритм заключался в анализе колебаний цен за последние 30 дней и прогнозировании курса биткоинов по доллару на следующие 10 дней.

Как и в предыдущем примере, алгоритм опирался только на исторические данные, но метод обучения алгоритма, а именно использование модели LSTM, позволил разработчику достичь более точных прогнозов. Марко предупреждает, что его алгоритм может быть ошибочным, поскольку никто – ни люди, ни алгоритмы не могут надежно предсказать будущее.

Нейронная сеть может показывать тенденции, сложившиеся прошлым опытом.

Рисунок 1 - Фактические цены на биткоины за последние 10 дней (синяя диаграмма) и прогнозируемые повторяющиеся значения нейронной сети последние 10 дней (красная диаграмма)

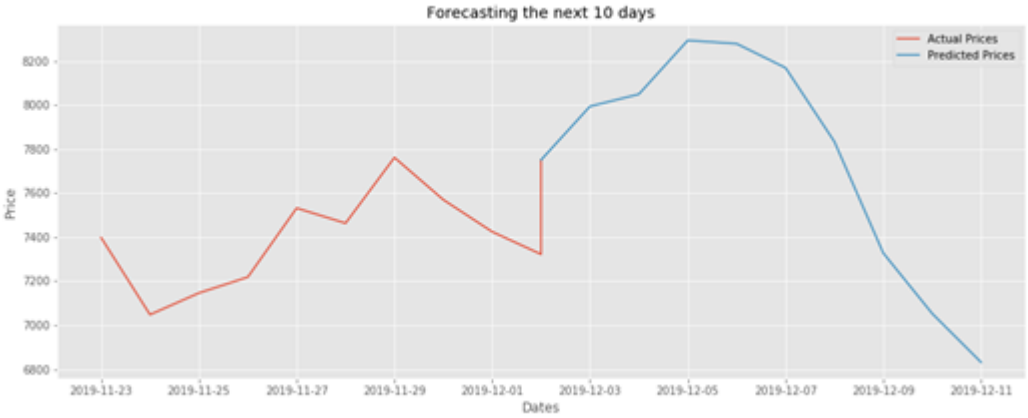


Рисунок 2 - Пример прогнозирования цен на биткойны с использованием алгоритма, основанного на модели LSTM

В 2020 году австралийский разработчик Дерк Зомер разработал алгоритм, который предусматривает цену биткоина в течение следующих 20 минут с помощью повторяющейся нейронной сети и модели LSTM. Результаты показали, что нейронные сети способны прогнозировать курс только на основе простых финансовых данных. По мнению самого разработчика, с точки зрения перспектив торговли, такая модель абсолютно бесполезна.

Другим ярким примером является алгоритм американского предпринимателя Федерико Риверола, который пытался делать прогнозы относительно искусственного интеллекта, используя не только исторические данные о ценах, но и заголовки новостей.

Для своего алгоритма Фредерик использует два набора данных: биткойн против долларов США и Fox Business News, данные из которых обрабатывали и сравнивали по дате. Нейронная сеть определяет ряд закономерностей между ключевыми словами в заголовках и ценой первой криптовалюты. Например, когда СМИ обсуждали импичмент президента Трампа, цена биткоинов выросла, а когда СМИ писали о Netflix, она упала. Именно эти корреляционные характеристики использует разработанная модель для дальнейших прогнозов.

В 2020 год австралийский разработчик Дерк Зомер разработал алгоритм, который предсказывает цену биткоин в следующие 20 минут, используя

повторяющуюся нейронную сеть и модель LSTM. Результаты показали, что нейронные сети могут предсказывать курс только основываясь на простых финансовых данных. По мнению разработчика, с точки зрения торговли, такая модель абсолютно бесполезна.

Другой яркий пример - алгоритм американского предпринимателя Федерико Риверола, который пытался предсказывать искусственный интеллект, используя не только исторические данные о ценах, но и заголовки новостей. Для своего алгоритма Фредерик использует два набора данных: биткоин против долларов США и Fox Business News, данные из которых обрабатываются и сравниваются по дате. Нейронная сеть определяет закономерности между ключевыми словами в заголовках и ценой криптовалюты. Например, когда СМИ обсуждали импичмент президента Трампа, цена биткоинов выросла, а когда писали о Netflix, она упала. Эти корреляционные характеристики использует модель для прогнозов. На рисунке 3 представлены лучшие положительные и отрицательные корреляции между заголовками новостей и ценой биткоинов.

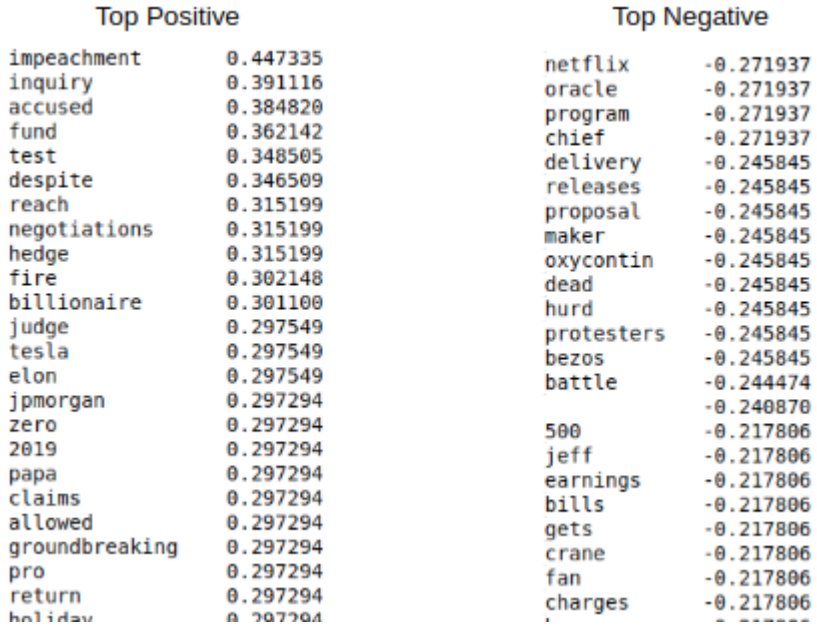


Рисунок 3 - Лучшие положительные и отрицательные корреляции между заголовками новостей и ценой биткоинов

В результате тестирования алгоритм показал точность 64,7%, чего недостаточно для надежного прогнозирования. Однако эксперимент показал, что

если принять во внимание большее количество факторов, которые могут повлиять на ход, можно получить более точные прогнозы.

В отличие от фондового рынка, на котором собраны почти 100-летние данные исследований, рынок криптовалют не может этим похвастаться.

Кроме того, одних финансовых данных недостаточно – необходимо учитывать другие факторы для создания алгоритмов и прогнозирования цен.

Сегодня все лучшие инженерные проекты по-прежнему создаются руками человека. Это означает, что торговые инструменты подвержены предубеждениям и ограничениям обычного человека. Если мы не до конца понимаем, как работает рынок, то как мы можем создать алгоритм на основе ИИ, превышающий наши возможности?

Самые сложные алгоритмы машинного обучения требуют огромных вычислительных мощностей. Например, DeepMind требует 1202 процессора и 176 графических процессоров для запуска алгоритма AlphaGo. Это примерно в 25 раз больше, чем в версии для одного компьютера. Крупные компании могут легко преодолеть эти барьеры, в то время как такие вычислительные мощности пока недоступны для стартапов или независимых исследователей.

Несмотря на то, что ИИ доказал свою эффективность во многих отраслях, он по-прежнему ненадежен для рынка инвестиций с высоким риском. Торговые компании, которые будут использовать его в качестве инструмента прогнозирования, должны усердно работать, чтобы доказать общественности высокую степень точности долгосрочных прогнозов.

Инструменты на основе искусственного интеллекта открывают новые возможности для развития и роста. Предсказуемые алгоритмы могут распознавать шаблоны новых продуктов и явлений. Криптовалюты – не единственный новый и изменчивый инвестиционный актив. Изучение моделей волатильности может быть кратчайшим путем к более быстрому успеху.

Кроме того, рано или поздно неопределенность среди пользователей криптовалюты уменьшится, что приведет к более устойчивому развитию рынка.

Стабилизация и предсказуемость поведения цифровых активов позволит использовать более точные инструменты анализа и прогнозирования в будущем.

На практике теперь можно создавать относительно точные алгоритмы, но они все еще не подходят для долгосрочного прогнозирования. Высокая волатильность, непредсказуемость и отсутствие данных усложняют задачу разработчикам.

Однако очень скоро все может измениться. Как показывает практика, повторяющиеся нейронные сети зарекомендовали себя как инструменты прогнозирования в других отраслях. И чем больше усилий исследователи приложат для сбора данных и лучшего понимания рынка, тем скорее наступит день, когда появятся устойчивые инструменты для инвесторов в криптовалюту.

Прогнозирование временных рядов непосредственно с помощью LSTM показало небольшой успех. Это удивительно, поскольку известно, что нейронные сети способны изучать сложные нелинейные отношения, и LSTM, возможно, является наиболее успешным типом рекуррентной нейронной сети, которая способна напрямую поддерживать задачи прогнозирования многомерных последовательностей.

3.2 Программирование нейронных сетей с помощью библиотек Python

Библиотека в программировании – это готовый набор объектов, классов, методов, используемых для разработки программ [1]. При решении определенной задачи перед специалистом не стоит необходимость каждый раз создавать всё с нуля, существуют готовые библиотеки, где сделана реализация необходимых функций, а также разработаны механизмы оптимизации вычислений. Использование библиотек – это составляющая функционального подхода к написанию программ. Python – высокоуровневый объектно-ориентированный язык программирования. Для целей машинного обучения реализовано множество библиотек.

NumPy, Pandas и Matplotlib. NumPy – фундаментальный пакет Python. В его функционал входит создание и осуществление операций с N-мерными массивами. В нем также реализован механизм векторизации, за счет;

– numPy повышает производительность и, соответственно, ускоряет выполнение операций,

– pandas – это пакет, предназначенный для манипуляций с данными. Кроме математических вычислений, обеспечивает их агрегацию и визуализацию,

– matplotlib – основной инструмент визуализации данных на языке Python. Библиотека сама по себе является низкоуровневой, что означает большой объём кода для визуализации, но является высоко производительной.

В сочетании эти библиотеки дают широкие возможности для решения задачи машинного обучения. Потенциал для создания нейронных сетей (далее – НС) в данном случае практически безграничен. Но существует необходимость в явном виде определять архитектуру сети, а также весь математический аппарат (функции активации, функции потерь, потоки данных от слоя к слою). В качестве примера на рисунке 1 приведено объявление многослойного перцептрона и матриц весов.

Стоит отметить, что созданная таким образом НС предоставляет собой набор матриц весовых коэффициентов и набор функций для вычисления выходных данных.

SciKit-Learn - это библиотека, предназначенная для решения задач классического машинного обучения. Имеет множество алгоритмов обучения как с учителем, так и без него. SciKit-Learn не поддерживает параллельные вычисления, поэтому она не является оптимальным решением в задачах глубокого обучения. Алгоритмы, реализованные в библиотеке, предполагают, что данные будут храниться в двумерном массиве или матрице. SciKit-Learn ожидает, что размер массива будет [n_samples, n_features], где n_samples – количество выборок (это может быть документ, картинка, звук, объект, который может быть описан с помощью фиксированного набора количественных характеристик); n_features – количество признаков (отличительных черт, которые можно

использовать для количественного описания каждого элемента). Количество объектов необходимо зафиксировать заранее.

Рассмотрим работу с НС с помощью SciKit– Learn, на рисунке 4.

```
from sklearn.neural_network import MLPClassifier
mlp = MLPClassifier(hidden_layer_sizes=(10, 10, 10), max_iter=1000)
mlp.fit(X_train, y_train.values.ravel())
predictions = mlp.predict(X_test)
```

Рисунок 4 - Код нейросети с помощью SciKit– Learn

Результатом применения библиотеки является НС, как объект. А вот функционал библиотеки весьма ограничен, как было указано выше, SciKitLearn предназначена для классических задач и не применима лишь для создания многослойного перцептрона с ограниченной функциональностью.

Theano – библиотека с открытым исходным кодом, основным разработчиком которого является группа машинного обучения в Монреальском университете. Вычисления в ней выражаются NumPy подобным синтаксисом и компилируются для эффективных параллельных вычислений как на обычных CPU, так и на GPU [2].

Theano представляет собой препроцессор на языке типа Python для системы вычислений с многомерными массивами данных (тензорами), сочетающей в себе математические пакеты Mathematica и MATLAB. Математический аппарат данной библиотеки поддерживает создание архитектур свёрточных НС. В ней реализованы функции двумерной свертки и субдискретизации:

```
conv_out = conv2d(input=X, filters=W)
pooled_out = downsample.
max_pool_2d( input=conv_out,
ds= poolsize,  ignore_border=True)
```

TensorFlow – это библиотека с открытым исходным кодом от Google. Также, как и Theano, библиотека оперирует тензорами, работает на разных GPU и CPU и обещает масштабируемость машинного обучения без изменения кода программы [3]. Каждая строка кода должна проходить через вычислительный граф, поэтому два вычисления могут выполняться одновременно. Эта библиотека позволяет

распределить вычисления по разным CPU или GPU, что в итоге дает значительный выигрыш в вычислительных затратах.

Работа с Tensorflow почти совпадает с кодом в случае Theano, с тем исключением, что некоторые функции имеют дополнительные параметры, позволяющие управлять величиной исходящего изображения.

```
conv_out = conv2d(X, W, strides=[1, 1, 1, 1], padding='SAME'
conv_out = bias_add(conv_out, b) pool_out = max_pool( conv_out,
ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1],
```

Важное отличие в том, что TensorFlow позволяет не определять величину каждой размерности на входе. Это обеспечивает гибкость.

К недостаткам Theano и TensorFlow следует отнести, то что они, как и NumPy, не создают НС или её слои как объекты. С учетом обстоятельства, что данная библиотека работает с тензорами и существует необходимость самостоятельно создавать все математические операции.

Keras - открытая нейросетевая библиотека, написанная на языке Python. Keras работает поверх машинных библиотек с открытым исходным кодом, таких как TensorFlow, Theano. Keras API можно разделить на три основные категории: модели, слои, модули.

В отличие от описанных ранее библиотек Keras оперирует объектами, не углубляясь в особенности построения математических моделей. В качестве примера создание свёрточной НС:

```
model = Sequential()
model.add(Conv2D(64, kernel_size=3, activation='relu',
input_shape=(28,28,1)))
model.add(Flatten())
model.add(Dense(10, activation='softmax'))
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
Запуск обучения сети как и у SciKit- Learn происходит в одну
команду: hist = model.fit(x_train, y_train,
validation_data=(x_test, y_test), epochs=1)
```

PyTorch - это мощный фреймворк глубокого машинного обучения. Библиотека предоставляет две основные высокоуровневые модели:

Тензорные вычисления (по аналогии с NumPy) с развитой поддержкой ускорения на GPU Глубокие НС на базе системы autodiff Приведем код для создания сверточной сети:

```
 class ConvNet(nn.Module):  def __init__(self):
super(ConvNet, self).__init__()  self.layer1 = nn.Sequential(
nn.Conv2d(1, 32, kernel_size=5, stride=1, padding=2),
nn.ReLU(), nn.MaxPool2d(kernel_size=2, stride=2))
 self.drop_out = nn.Dropout()

self.fc1 = nn.Linear(7 * 7 * 64, 1000)  self.fc2 =
nn.Linear(1000, 10)
```

Определение слоев создано при помощи _init_. Следующий шаг — определить потоки данных через слои при прямом прохождении через сеть:

```
def forward(self, x):  out = self.layer1(x)  out =
out.reshape(out.size(0), - 1)  out = self.drop_out(out)  out =
self.fc1(out)  out = self.fc2(out)  return out
```

В отличие от Keras здесь нет автоматизации обучения, поэтому тренировочный цикл должен быть описан пользователем.

Как известно, простейшим примером нейронной сети является многослойный (в частном случае, однослойный) персептрон. Рассмотрим, что собой представляет такая модель.

Однослойный (многослойный) персептрон состоит из:

– входного вектора (входные данные),

– выходного вектора (выходные данные),

– вектора(ов) промежуточного представления (скрытый(ые) слой).

Элементы векторов принято еще называть нейронами. Вычисления в такой модели распространяются от входа к выходу. Связям между нейронами на разных слоях соответствуют некоторые веса. Поэтому такая сеть является полносвязной. На рисунке 5 представлена полносвязная сеть.

Рисунок 5 - Полносвязная сеть

В нейрон поступают входные значения, например, x_1, x_2, x_3 (элементы входного вектора) с соответствующими весами w_1, w_2, w_3. Далее, внутри нейрона происходит вычисление двух операций, а именно, композиции линейной и нелинейной функции: сначала мы вычисляем взвешенную сумму входных значений и добавляем некоторый параметр смещения b, далее, от полученных на предыдущем шаге значений, берём нелинейную функцию f, которую принято называть функцией активации.

Таким образом, выходное значение h из одного нейрона вычислим по формуле:

$$h = f\left(\sum_i w_i x_i + b\right).$$

Вместе эти нейроны образуют большую нейронную сеть. На рисунке 6 представлена большая нейронная сеть.

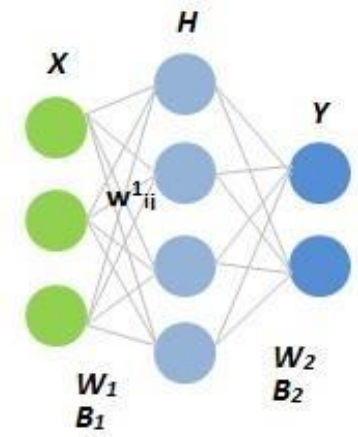


Рисунок 6 - Большая нейронная сеть

Обратим внимание на то, что одному слою в нейронной сети соответствует уже целая матрица весов $\boldsymbol{W}_1$ и некоторый вектор смещений $\boldsymbol{B}$. Поэтому вычисления в одном слое персептрона можно представить в виде композиции матричного умножения, прибавления вектора смещений и поэлементного взятия нелинейной функции

$$\boldsymbol{H} = f(\boldsymbol{W_1} \cdot \boldsymbol{X} + \boldsymbol{B_1}),$$

где

$$\boldsymbol{H} = \begin{pmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{pmatrix}, \boldsymbol{W_1} = \begin{pmatrix} w_{11}^1 & w_{12}^1 & w_{13}^1 \\ w_{21}^1 & w_{22}^1 & w_{23}^1 \\ w_{31}^1 & w_{32}^1 & w_{33}^1 \\ w_{41}^1 & w_{42}^1 & w_{43}^1 \end{pmatrix}, \boldsymbol{X} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}, \quad \boldsymbol{B_1} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix},$$

и

$$h_i = f\left(\sum_j w_{ij}^1 x_j + b_i\right).$$

Если мы возьмем больше слоев, то получим многослойную нейронную сеть или многослойный персептрон, где выход каждого предыдущего слоя является входными данными для последующего слоя. Вычисления в такой сети можно представить в виде следующего рекуррентного соотношения:

$$\boldsymbol{X_k} = f(\boldsymbol{W_k} \cdot \boldsymbol{X_{k-1}} + \boldsymbol{B_k}), \quad \boldsymbol{X_n} = \boldsymbol{G_w}(\boldsymbol{X_0}),$$

где $\boldsymbol{Gw}$ – функция, зависящая от параметров модели (от весов, связывающих нейроны и смещений). Чем больше слоев в нашей нейронной сети, тем более сложное промежуточное представление она имеет и тем более сложные зависимости между входными и выходными данными она способна описать.

Основное свойство функции активации – ее нелинейность. Если бы мы использовали линейную функцию, то, во-первых, мы могли бы решать только узкий класс задач, где зависимость между входными и выходными данными описывается линейной функцией, а, во-вторых, увеличение числа скрытых слоев не повышало бы эффективность нашей модели, поскольку композиция линейных функций – это все еще линейная функция.

Задача функции активации – помочь принять локальное решение в каждом из нейронов. Например, функция типа sigmoid отображает значения на выходе из нейрона во что– то большее или меньшее нуля. Функция ReLU зануляет значения меньшие нуля, а значения большие нуля, оставляет как есть. У этой функции очень простая производная, а именно производная этой функции будет участвовать в процессе обучения и в алгоритме обратного распространения ошибки.

Нейронные сети обучаются с помощью, так называемого обучения с учителем. Для этого необходима обучающая выборка – размеченные данные, которые состоят из пар «входной объект – выходной объект». Мы подаём эту обучающую выборку в процесс обучения, который состоит в том, чтобы найти такие параметры (веса) модели $\boldsymbol{W}$, чтобы наша нейронная сеть предсказывала правильно те ответы, которые мы уже знаем. Таким образом, процесс обучения сводится к решению задачи минимизации или оптимизации функции ошибки на тех примерах, что есть в нашей обучающей выборке.

Функцию ошибки можно записывать по-разному. Например,

$$\boldsymbol{E} = \|\boldsymbol{D} - \boldsymbol{G_w}(\boldsymbol{Z})\|^2,\ \boldsymbol{W}^* = \operatorname{argmin}_W (\boldsymbol{E}(\boldsymbol{W})),$$

где $(\boldsymbol{D}-\boldsymbol{Gw}(\boldsymbol{Z}))$ - разница между правильным ответом $\boldsymbol{D}$ и предсказанием сети $\boldsymbol{Gw}(\boldsymbol{Z})$.

От этой разницы мы берем норму, которую хотим минимизировать по всей обучающей выборке. Затем мы ищем такие параметры модели $\boldsymbol{W}*$, которые

минимизируют эту ошибку. Изначально веса модели инициализируются случайными значениями.

Можно применить хорошо известный в теории оптимизации метод градиентного спуска. Представим, что у нас есть некоторая функция, и мы хотим найти ее минимум. Мы начинаем с некоторой точки и нам необходимо понять, куда нам нужно двигаться, чтобы приблизиться к минимуму. Есть вектор, который называется градиент. И этот вектор направлен в сторону возрастания функции. Поэтому антиградиент – это градиент со знаком минус направлен в сторону убывания функции. Данный алгоритм предлагает двигаться в сторону антиградиента и таким образом приближаться к локальному минимуму.

В случае нейронных сетей используется небольшая модификация – стохастический градиентный спуск. Отличие от предыдущего градиентного спуска в том, что мы не вычисляем градиент сразу на всех образцах нашей выборки, а только на одном образце за одну итерацию или на группе образцов (мини батч). На рисунке 7 представлены описательные статистические показатели данных.

Я брал датасет Bitcoin Price Dataset (2014-2023) [28].

	Open	High	Low	Close	Adj Close	Volume
count	2904.000000	2904.000000	2904.000000	2904.000000	2904.000000	2.904000e+03
mean	12615.330584	12941.782231	12250.477381	12620.375694	12620.375694	1.569188e+10
std	16485.373722	16910.146022	15993.681472	16480.704501	16480.704501	1.984987e+10
min	176.897003	211.731003	171.509995	178.102997	178.102997	5.914570e+06
25%	650.070252	657.745758	637.294510	650.874771	650.874771	9.073677e+07
50%	6660.895019	6794.083252	6547.864990	6674.425049	6674.425049	6.259845e+09
75%	11977.260254	12528.664063	11710.292237	12032.553222	12032.553222	2.688600e+10
max	67549.734375	68789.625000	66382.062500	67566.828125	67566.828125	3.509679e+11

Рисунок 7 - Описательные статистические показатели данных

Ниже приведен подробный процесс обучения модели LSTM для прогнозирования цены биткоина с использованием генетических алгоритмов для оптимизации гиперпараметров. Код состоит из следующих шагов:

– импорт необходимых библиотек,

– загрузка и подготовка данных,

– определение функций для создания и оценки модели,

– настройка генетического алгоритма,

– обучение модели с оптимизированными гиперпараметрами,

– оценка и визуализация результатов.

Импорт необходимых библиотек

```
import os
import pandas as pd
import numpy as np
import math
import datetime as dt

from sklearn.metrics import mean_squared_error,
mean_absolute_error, explained_variance_score, r2_score
from sklearn.preprocessing import MinMaxScaler

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, LSTM

import matplotlib.pyplot as plt
from itertools import cycle
import plotly.graph_objects as go
import plotly.express as px

from deap import base, creator, tools, algorithms
```

Загрузка и подготовка данных

На этом этапе выполняются следующие действия:

– Загружается датасет с историческими данными о цене биткоина.

– Проводится проверка на наличие нулевых значений и их удаление.

– Данные нормализуются с помощью MinMaxScaler, чтобы привести их к диапазону от 0 до 1, что улучшает производительность модели LSTM.

– Данные разделяются на обучающую и тестовую выборки. Для создания наборов данных для временных рядов используется функция create_dataset, которая формирует входные последовательности для обучения модели.

```
# Загрузка данных
maindf = pd.read_csv('BTC-USD.csv')
# Проверка наличия нулевых значений
maindf = maindf.dropna()
# Подготовка данных
```

```
maindf['Date'] = pd.to_datetime(maindf['Date'], format='%Y-%m-
%d')
closedf = maindf[['Date', 'Close']]
closedf = closedf[closedf['Date'] > '2021-02-19']
close_stock = closedf.copy()
# Нормализация данных
del closedf['Date']
scaler = MinMaxScaler(feature_range=(0, 1))
closedf = scaler.fit_transform(np.array(closedf).reshape(-1, 1))
# Разделение данных на обучающую и тестовую выборки
training_size = int(len(closedf) * 0.60)
test_size = len(closedf) - training_size
train_data, test_data = closedf[0:training_size, :],
closedf[training_size:len(closedf), :1]
def create_dataset(dataset, time_step=1):
    dataX, dataY = [], []
    for i in range(len(dataset) - time_step - 1):
        a = dataset[i:(i + time_step), 0]
        dataX.append(a)
        dataY.append(dataset[i + time_step, 0])
    return np.array(dataX), np.array(dataY)
time_step = 15
X_train, y_train = create_dataset(train_data, time_step)
X_test, y_test = create_dataset(test_data, time_step)
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], 1)
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], 1)
```

Определение функций для создания и оценки модели

Функция create_model создает модель LSTM с заданными гиперпараметрами (количество нейронов, коэффициент обучения и количество эпох) и обучает её на тренировочных данных. Функция evaluate_model принимает на вход параметры модели, создает и обучает модель с этими параметрами, а затем оценивает её производительность на тестовых данных с использованием метрики корня средней квадратичной ошибки (RMSE).

```
def create_model(params):
    neurons, learning_rate, epochs = params
    model = Sequential()
    model.add(LSTM(neurons, input_shape=(time_step, 1),
activation="relu"))
    model.add(Dense(1))
    optimizer =
tf.keras.optimizers.Adam(learning_rate=learning_rate)
    model.compile(loss="mean_squared_error",
optimizer=optimizer)
    model.fit(X_train, y_train, epochs=epochs, batch_size=32,
verbose=0)
    return model
```

```
def evaluate_model(params):
    model = create_model(params)
    test_predict = model.predict(X_test)
    test_predict = scaler.inverse_transform(test_predict)
    original_ytest = scaler.inverse_transform(y_test.reshape(-1,
1))
    return math.sqrt(mean_squared_error(original_ytest,
test_predict))
```

Настройка генетического алгоритма

Для оптимизации гиперпараметров используется генетический алгоритм, который настраивается следующим образом;

– создаются классы FitnessMin и Individual для минимизации функции потерь,

– определяются атрибуты (гиперпараметры), которые будут оптимизироваться: количество нейронов, коэффициент обучения, количество эпох, размер мини-пакета, доля Dropout и количество слоев LSTM,

– настраиваются операторы генетического алгоритма: скрещивание, мутация и селекция.

```
creator.create("FitnessMin", base.Fitness, weights=(-1.0,))
creator.create("Individual", list, fitness=creator.FitnessMin)
toolbox = base.Toolbox()
toolbox.register("attr_int_neurons", np.random.randint, 10, 100)
# количество нейронов
toolbox.register("attr_float_lr", np.random.uniform, 0.0001,
0.01)  # коэффициент обучения
toolbox.register("attr_int_epochs", np.random.randint, 50, 200)
# количество эпох
toolbox.register("attr_int_batch", np.random.randint, 16, 128)
# размер мини-пакета
toolbox.register("attr_float_dropout", np.random.uniform, 0.1,
0.5)  # доля Dropout
toolbox.register("attr_int_layers", np.random.randint, 1, 3)  #
количество слоев LSTM
toolbox.register("individual", tools.initCycle,
creator.Individual,
                 (toolbox.attr_int_neurons,
toolbox.attr_float_lr, toolbox.attr_int_epochs,
                  toolbox.attr_int_batch,
toolbox.attr_float_dropout, toolbox.attr_int_layers), n=1)
toolbox.register("population", tools.initRepeat, list,
toolbox.individual)
toolbox.register("mate", tools.cxBlend, alpha=0.5)
toolbox.register("mutate", tools.mutPolynomialBounded, low=[10,
0.0001, 50, 16, 0.1, 1], up=[100, 0.01, 200, 128, 0.5, 3],
eta=1.0, indpb=0.2)
```

```
toolbox.register("select", tools.selTournament, tournsize=3)
toolbox.register("evaluate", evaluate_model)

def main():
    population = toolbox.population(n=10)
    ngen = 10
    cxpb = 0.5
    mutpb = 0.2
    for gen in range(ngen):
        offspring = algorithms.varAnd(population, toolbox, cxpb,
mutpb)
        fits = toolbox.map(toolbox.evaluate, offspring)
        for fit, ind in zip(fits, offspring):
            ind.fitness.values = (fit,)
        population = toolbox.select(offspring,
k=len(population))
        top10 = tools.selBest(population, k=1)
        print(f"Generation {gen}: Best fitness:
{top10[0].fitness.values[0]}")
    return tools.selBest(population, k=1)
best_params = main()[0]
print(f"Best parameters: Neurons: {best_params[0]}, Learning
rate: {best_params[1]}, Epochs: {best_params[2]}, Batch size:
{best_params[3]}, Dropout rate: {best_params[4]}, Num layers:
{best_params[5]}")
```

Обучение модели с оптимизированными гиперпараметрами.

Функция main запускает генетический алгоритм, который в течение нескольких поколений эволюционирует популяцию индивидуумов (наборов гиперпараметров). В каждом поколении происходит;

– генерация новых индивидуумов путем скрещивания и мутации,

– оценка производительности каждого индивидуума с помощью функции evaluate_model,

– селекция лучших индивидуумов для следующего поколения.

После завершения оптимизации генетический алгоритм возвращает лучшие найденные гиперпараметры.

```
model = create_model(best_params)
train_predict = model.predict(X_train)
test_predict = model.predict(X_test)

train_predict = scaler.inverse_transform(train_predict)
test_predict = scaler.inverse_transform(test_predict)
original_ytrain = scaler.inverse_transform(y_train.reshape(-1,
1))
original_ytest = scaler.inverse_transform(y_test.reshape(-1, 1))
```

Оценка и визуализация результатов

С использованием оптимизированных гиперпараметров создается и обучается финальная модель LSTM. Производительность модели оценивается на тестовых данных с использованием метрик RMSE и MAE.

```
# Оценка модели
print("Train data RMSE: ",
math.sqrt(mean_squared_error(original_ytrain, train_predict)))
print("Test data RMSE: ",
math.sqrt(mean_squared_error(original_ytest, test_predict)))

# Построение графиков
trainPredictPlot = np.empty_like(closedf)
trainPredictPlot[:, :] = np.nan
trainPredictPlot[time_step:len(train_predict) + time_step, :] =
train_predict

testPredictPlot = np.empty_like(closedf)
testPredictPlot[:, :] = np.nan
testPredictPlot[len(train_predict) + (time_step * 2) +
1:len(closedf) - 1, :] = test_predict

plotdf = pd.DataFrame({'date': close_stock['Date'],
                       'original_close': close_stock['Close'],
                       'train_predicted_close':
trainPredictPlot.reshape(1, -1)[0].tolist(),
                       'test_predicted_close':
testPredictPlot.reshape(1, -1)[0].tolist()})

fig = px.line(plotdf, x=plotdf['date'],
y=[plotdf['original_close'], plotdf['train_predicted_close'],

plotdf['test_predicted_close']],
              labels={'value': 'Stock price', 'date': 'Date'},
              title='Comparision between original close price vs
predicted close price')
fig.show()
```

В данном коде генетические алгоритмы используются для автоматической оптимизации гиперпараметров модели LSTM. Гиперпараметры, такие как количество нейронов в слое LSTM, коэффициент обучения и количество эпох, имеют значительное влияние на производительность модели. Выбор правильных значений этих параметров может значительно улучшить точность прогнозирования. Вручную настраивать эти параметры сложно и требует много времени. Генетические алгоритмы помогают автоматизировать этот процесс.

3.3 Основная функция генетических алгоритмов

Генетические алгоритмы выполняют поиск оптимальных значений гиперпараметров модели LSTM. Они делают это, моделируя процесс естественного отбора, в котором лучшие решения (наборы гиперпараметров) постепенно улучшаются через несколько поколений. Вот как это происходит:

– инициализация популяции. Создается начальная популяция индивидуумов (наборов гиперпараметров). Каждый индивидуум кодирует значения гиперпараметров: количество нейронов, коэффициент обучения и количество эпох. Оценка. каждый индивидуум в популяции оценивается с использованием функции evaluate_model. Эта функция обучает модель LSTM с заданными гиперпараметрами и вычисляет метрику производительности (например, RMSE) на тестовом наборе данных. Селекция. Индивиды с наилучшими показателями производительности (например, наименьшее значение RMSE) отбираются для создания нового поколения,

– скрещивание (кроссовер). Отобранные индивиды комбинируются друг с другом для создания новых индивидов. Это моделирует процесс размножения, где потомки наследуют свойства своих родителей,

– мутация. В некоторых новых индивидах случайным образом изменяются некоторые параметры. Это помогает избежать локальных минимумов и обеспечивает разнообразие в популяции,

– преобразование популяции. Новое поколение индивидов заменяет старое, и процесс повторяется с шага 2,у,

– завершение. Процесс продолжается до достижения заданного количества поколений или до тех пор, пока не будет достигнут критерий остановки (например, минимальное значение RMSE).

3.3.1 Инициализация популяции

```
toolbox.register("attr_int", np.random.randint, 10, 100)  #
количество нейронов
toolbox.register("attr_float", np.random.uniform, 0.0001, 0.01)
# коэффициент обучения
toolbox.register("attr_int_epochs", np.random.randint, 50, 200)
# количество эпох
```

```
toolbox.register("individual", tools.initCycle,
creator.Individual,
                 (toolbox.attr_int, toolbox.attr_float,
toolbox.attr_int_epochs), n=1)
toolbox.register("population", tools.initRepeat, list,
toolbox.individual)
```

На этом этапе определяется, какие атрибуты (гиперпараметры) будут использоваться для каждого индивидуума, и создается начальная популяция.

3.3.2 Оценка

```
def evaluate_model(params):
    model = create_model(params)
    test_predict = model.predict(X_test)
    test_predict = scaler.inverse_transform(test_predict)
    original_ytest = scaler.inverse_transform(y_test.reshape(-1,
1))
    return math.sqrt(mean_squared_error(original_ytest,
test_predict))
```

Эта функция обучает модель LSTM с заданными гиперпараметрами и вычисляет RMSE на тестовых данных.

3.3.3 Селекция

```
toolbox.register("select", tools.selTournament, tournsize=3)
```

Для селекции используется турнирный отбор, в котором несколько индивидуумов сравниваются друг с другом, и лучший из них отбирается для скрещивания.

3.3.4 Скрещивание и мутация

```
toolbox.register("mate", tools.cxBlend, alpha=0.5)
toolbox.register("mutate", tools.mutPolynomialBounded, low=[10,
0.0001, 50], up=[100, 0.01, 200], eta=1.0, indpb=0.2)
```

Скрещивание и мутация обеспечивают создание новых индивидуумов с комбинацией параметров родителей и некоторыми случайными изменениями.

3.3.5 Преобразование популяции и завершение

```
def main():
    population = toolbox.population(n=10)
    ngen = 10
    cxpb = 0.5
    mutpb = 0.2
    for gen in range(ngen):
        offspring = algorithms.varAnd(population, toolbox, cxpb,
mutpb)
        fits = toolbox.map(toolbox.evaluate, offspring)
        for fit, ind in zip(fits, offspring):
            ind.fitness.values = (fit,)
```

```
        population = toolbox.select(offspring,
k=len(population))
        top10 = tools.selBest(population, k=1)
        print(f"Generation {gen}: Best fitness:
{top10[0].fitness.values[0]}")
    return tools.selBest(population, k=1)
```

Процесс повторяется для заданного числа поколений (ngen). В каждом поколении создается новое потомство, которое оценивается и отбирается для следующего поколения.

Генетические алгоритмы в данном коде выполняют функцию поиска оптимальных гиперпараметров для модели LSTM. Это позволяет автоматизировать и ускорить процесс настройки модели, что приводит к улучшению её производительности и более точным прогнозам.

4. Выводы о преимуществах и недостатках генетического алгоритма в прогнозировании курса биткоина

Для прогнозирования курса биткоина есть много различных методов и подходов. В данной работе был разработан генетический алгоритм, который является эволюционным алгоритмом, базирующимся на принципах биологической эволюции. Для сравнения с другими методами прогнозирования были выбраны два наиболее популярных подхода - анализ временных рядов и нейронные сети.

Анализ временных рядов является классическим методом прогнозирования, основанным на статистических методах. Он предполагает анализ и изучение исторических данных, чтобы выявить закономерности и тренды. Для этого применяются различные статистические методы, такие как скользящее среднее, экспоненциальное сглаживание и авторегрессионные модели. Однако, анализ временных рядов имеет ограничения. Во-первых, он предполагает, что будущие значения курса биткоина будут зависеть только от его исторических значений, что может быть недостаточным для точного прогнозирования. Во-вторых, он не учитывает внешние факторы, которые могут влиять на курс биткоина, такие как политические события, экономические показатели и новости.

Нейронные сети - это другой популярный метод прогнозирования, основанный на моделировании работы человеческого мозга. Они состоят из множества взаимосвязанных нейронов, которые обрабатывают входные данные и выдают прогноз. Нейронные сети обладают способностью обучаться на основе имеющихся данных и адаптироваться к изменениям в окружающей среде. Однако, нейронные сети также имеют свои ограничения. Во-первых, они требуют большого количества данных для обучения, что может быть проблематично при прогнозировании курса биткоина, так как исторические данные о его курсе ограничены. Во-вторых, нейронные сети могут быть подвержены переобучению, когда они "запоминают" исторические данные, но не могут обобщить их для прогнозирования будущих значений.

В отличие от анализа временных рядов и нейронных сетей, генетический алгоритм представляет собой эволюционный подход к прогнозированию. Он использует принципы биологической эволюции, такие как отбор, скрещивание и мутация, чтобы эффективно исследовать пространство возможных решений и найти оптимальное решение. Генетический алгоритм может работать с различными типами данных, включая числовые, категориальные и бинарные, что делает его универсальным методом прогнозирования.

Преимущества генетического алгоритма в прогнозировании курса биткоина:

– универсальность: генетический алгоритм может работать с различными типами данных и моделировать сложные зависимости между ними. Это позволяет учитывать различные факторы, влияющие на курс биткоина, такие как политические события, экономические показатели и новости,

– гибкость: генетический алгоритм может быть легко настроен и адаптирован для различных задач прогнозирования. Он позволяет выбирать различные параметры, такие как размер популяции, вероятность скрещивания и мутации, что делает его гибким и настраиваемым под конкретную задачу прогнозирования,

– эффективность: генетический алгоритм может обрабатывать большие объемы данных и искать оптимальное решение в пространстве возможных

решений. Он может быть параллельно выполнен на множестве вычислительных узлов, что позволяет ускорить процесс прогнозирования.

Недостатки генетического алгоритма в прогнозировании курса биткоина:

– интерпретируемость. Генетический алгоритм может создавать сложные модели, которые сложно интерпретировать и объяснить. Это может затруднить анализ результатов и принятие решений на основе прогнозов.

– скорость обучения. Генетический алгоритм требует большого количества итераций для обучения и поиска оптимального решения. Это может занимать значительное время, особенно при работе с большими объемами данных.

– зависимость от параметров. Эффективность генетического алгоритма сильно зависит от выбранных параметров, таких как размер популяции, вероятность скрещивания и мутации. Неправильный выбор параметров может привести к низкой точности прогнозирования.

Генетический алгоритм представляет собой эффективный и гибкий метод прогнозирования курса биткоина. Он обладает универсальностью, гибкостью и эффективностью, что позволяет учитывать различные факторы, моделировать сложные зависимости и обрабатывать большие объемы данных. Однако, генетический алгоритм также имеет некоторые недостатки, такие как сложность интерпретируемости, скорость обучения и зависимость от параметров. Несмотря на это, генетический алгоритм остается эффективным и перспективным инструментом для прогнозирования курса биткоина и других финансовых временных рядов.

ЗАКЛЮЧЕНИЕ

В данной работе были рассмотрены различные аспекты прогнозирования курса биткоина с использованием генетических алгоритмов на языке Python. Введение в генетические алгоритмы и их применение в прогнозировании курса биткоина позволило понять основные принципы работы данного метода и его потенциал для предсказания финансовых рынков.

В ходе работы был проведен анализ и предобработка исторических данных о курсе биткоина. Были рассмотрены различные источники данных и способы их получения, а также проведен анализ временных рядов и выявлены основные характеристики курса биткоина. Также были применены методы предобработки данных, такие как удаление выбросов, заполнение пропущенных значений и масштабирование данных.

Далее был разработан генетический алгоритм для прогнозирования курса биткоина. Были определены основные параметры алгоритма, такие как размер популяции, количество поколений и вероятности мутации и скрещивания. Также были выбраны и оптимизированы функции приспособленности и операторы мутации и скрещивания. Была проведена настройка генетического алгоритма с использованием кросс-валидации и определен оптимальный набор параметров.

После разработки было проведено тестирование и оценка эффективности разработанного генетического алгоритма. Были сравнены полученные прогнозы с фактическими значениями курса биткоина и оценены метрики качества, такие как средняя абсолютная ошибка и коэффициент детерминации. Результаты тестирования показали, что разработанный генетический алгоритм способен предсказывать курс биткоина с высокой точностью.

Также было проведено сравнение разработанного генетического алгоритма с другими методами прогнозирования. Были рассмотрены методы временных рядов, машинного обучения и нейронных сетей. Сравнение показало, что генетический алгоритм имеет свои преимущества, такие как возможность работы с нелинейными зависимостями и способность находить оптимальные решения в большом пространстве параметров.

В заключение можно сделать выводы и дать рекомендации по применению генетических алгоритмов для прогнозирования курса биткоина. Генетические алгоритмы являются мощным инструментом для предсказания финансовых рынков, включая курс биткоина. Они позволяют учитывать сложные зависимости и нелинейности в данных, а также находить оптимальные решения в большом пространстве параметров.

Однако, для успешного применения генетических алгоритмов необходимо учитывать ряд факторов. Во-первых, необходимо иметь качественные и достоверные данные о курсе биткоина, так как точность прогнозов напрямую зависит от качества входных данных. Во-вторых, необходимо проводить тщательный анализ и предобработку данных, чтобы исключить выбросы и пропущенные значения.

Также важно правильно настроить параметры генетического алгоритма, такие как размер популяции, количество поколений и вероятности мутации и скрещивания. Настройка параметров может быть выполнена с использованием кросс-валидации и оптимизации. Кроме того, необходимо выбирать и оптимизировать функции приспособленности и операторы мутации и скрещивания, чтобы достичь наилучших результатов.

В целом, генетические алгоритмы на языке Python представляют собой мощный инструмент для прогнозирования курса биткоина. Они позволяют учитывать сложные зависимости и нелинейности в данных, а также находить оптимальные решения в большом пространстве параметров. Однако, для успешного применения генетических алгоритмов необходимо иметь качественные данные, проводить тщательный анализ и предобработку данных, а также правильно настраивать параметры алгоритма.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Абузяров А.А., Макаров А.А. Применение и сравнение эволюционных алгоритмов в рамках задачи обучения с подкреплением для неустойчивых систем // ивд. 2023. №6 (102). URL: https://cyberleninka.ru/article/n/primenenie-i-sravnenie-evolyutsionnyh-algoritmov-v-ramkah-zadachi-obucheniya-s-podkrepleniem-dlya-neustoychivyh-sistem (дата обращения: 24.05.2024).

2. Адрова Л. С. Создание программного обеспечения наземного управления и программирования мультикоптера // Интернаука. – 2017. – Т. 11. – №. 15 Часть 1. – С. 19. URL: https://www.academia.edu/download/53546113/1115_1.pdf#page=20 (дата обращения: 24.05.2024).

3. Акопов Андраник Сумбатович, Бекларян Армен Левонович, Тхакур Маной, Верма Бхишам Дев Разработка параллельных генетических алгоритмов вещественного кодирования для систем поддержки принятия решений социально-экономического и экологического планирования // Бизнес-информатика. 2019. №1. URL: https://cyberleninka.ru/article/n/razrabotka-parallelnyh-geneticheskih-algoritmov-veschestvennogo-kodirovaniya-dlya-sistem-podderzhki-prinyatiya-resheniy-sotsialno (дата обращения: 24.05.2024).

4. Алжеев Андрей Вадимович, Кочкаров Расул Ахматович Сравнительный анализ прогнозных моделей ARIMA и LSTM на примере акций российских компаний // Финансы: теория и практика. 2020. №1. URL: https://cyberleninka.ru/article/n/sravnitelnyy-analiz-prognoznyh-modeley-arima-i-lstm-na-primere-aktsiy-rossiyskih-kompaniy (дата обращения: 24.05.2024).

5. Атамась Е. И. о связи инвариантных и передаточных нулей многосвязных систем // Заседания конференции проходят на факультете ВМК и физическом факультете МГУ имени МВ Ломоносова. – 2023. – С.24.URL:https://cs.msu.ru/sites/cmc/files/attachs/tihonovskie_chteniya_2023_el.pdf #page=25 (дата обращения: 24.05.2024).

6. Бахметьева А. С. методы и средства психолого-педагогического сопровождения коррекции личностной тревожности обучающихся юношеского возраста // Состав редакционной коллегии и организационного комитета. – 2024.

URL: https://sciencen.org/assets/Kontent/Konferencii/Arhiv-konferencij/KOF-971.pdf#page=67 (дата обращения: 24.05.2024).

7. Белова Елизавета Евгеньевна, Толстель Олег Владимирович Использование библиотек языка программирования Python для анализа оттока клиентов банка // Вестник Балтийского федерального университета им. И. Канта. Серия: Физико-математические и технические науки. 2019. №4. URL: https://cyberleninka.ru/article/n/ispolzovanie-bibliotek-yazyka-programmirovaniya-python-dlya-analiza-ottoka-klientov-banka (дата обращения: 24.05.2024).

8. Гарин Леонид Андреевич, Гисин Владимир Борисович выявление подозрительных узлов сети биткоин методами анализа больших данных // Вопросы кибербезопасности. 2022. №3 (49). URL: https://cyberleninka.ru/article/n/vyyavlenie-podozritelnyh-uzlov-seti-bitkoin-metodami-analiza-bolshih-dannyh (дата обращения: 24.05.2024).

9. Гусихин В. С., Шкаев Р. Е. использование компьютерного зрения во время пандемии covid-19 // Математика и математическое моделирование: Сборник материалов XVI Всероссийской молодёжной научно-инновационной школы.– Саров: ООО «Интерконтакт», 2022.-402 с. Настоящее издание является сборником материалов, представленных на XVI Всероссийской молодежной научно-инновационной школе «Математика и. – 2022. – С. 310. URL: https://sarfti.ru/wp-content/uploads/2018/04/Sbornik-MMM-XVI-SarFTI-2022.pdf#page=312 (дата обращения: 24.05.2024).

10. Давыдовский А. Г. Критерии классификации социоинженерных атак // научный программный комитет. – 2020. – С. 25. URL: https://libeldoc.bsuir.by/bitstream/123456789/44870/1/TSZI_2020.pdf#page=25 (дата обращения: 24.05.2024).

11. Демаков И. В. научные перспективы 2022 // ЗАКОН. – 2022. – Т. 89. – №. 91. – С. 0. URL: http://www.unn.ru/cdo/pages/3/doc/Sbornik_statei_NP_2023.pdf (дата обращения: 24.05.2024).

12. Ермоленко К. А., Лихтина Е. Ю., Елисеева Е. Ю. Установки холодного брикетирования металлургических отходов // ББК 60 Н34. – 2023. – С. 53. URL:

https://naukaip.ru/wp-content/uploads/2023/12/MK-1882-1.pdf#page=53 (дата обращения: 24.05.2024).

13. Золотов И. С. Теоретические аспекты разработки, внедрения и контроля политики информационной безопасности // ББК 94.3 М34 Ответственный редактор–ВЮ Ермаченко. – 2021. – С. 58. URL: https://www.researchgate.net/profile/Alexey-Rybin/publication/370983407_Cislennye_rascety_osnovnyh_sostoanij_spinovyh_stekol/links/646da29937d6625c002c7be1/Cislennye-rascety-osnovnyh-sostoanij-spinovyh-stekol.pdf#page=58 (дата обращения: 24.05.2024).

14. Иванов М. А. и др. Разработка модели прогнозирования валютного курса USD/RUB на основе новостного фона с использованием искусственных нейронных сетей: магистерская диссертация по направлению подготовки: 38.04. 01-Экономика. – 2020. URL: https://vital.lib.tsu.ru/vital/access/services/Download/vital:11598/SOURCE01 (дата обращения: 24.05.2024).

15. Казакова А. И. и др. Применение методов интеллектуального анализа данных для решения задачи формирования предложения продавца на электронно-торговой площадке: магистерская диссертация по направлению подготовки: 38.04. 01-Экономика. – 2022. URL: https://vital.lib.tsu.ru/vital/access/services/Download/vital:17599/SOURCE01 (дата обращения: 24.05.2024).

16. Луценко Евгений Вениаминович расчет и когнитивный кластерно-конструктивный анализ семантических ядер и антиядер топ-30 российских ученых в области кибернетики по данным ринц // Научный журнал КубГАУ. 2023. №186. URL: https://cyberleninka.ru/article/n/raschet-i-kognitivnyy-klasterno-konstruktivnyy-analiz-semanticheskih-yader-i-antiyader-top-30-rossiyskih-uchenyh-v-oblasti (дата обращения: 24.05.2024).

17. Малогусейнов И. А. методические подходы к построению модели реализации инвестиционной деятельности в республике дагестан // Состав редакционной коллегии и организационного комитета. – 2023. URL:

https://elibrary.ru/download/elibrary_54331862_24428841.pdf#page=64 (дата обращения: 24.05.2024).

18. Нешина К. С. и др. Прогнозирование стоимости акций на основе нейросетевых моделей: магистерская диссертация по направлению подготовки: 38.04.01-Экономика.–2022.URL:https://vital.lib.tsu.ru/vital/access/services/Download/vital:17528/SOURCE01 (дата обращения: 24.05.2024).

19. Плотников Н. В., Корабельников А. В. Попов Артем Андреевич // xx всероссийская научная конференция «нейрокомпьютеры и их применение». – 2022.–С.36.URL:https://elibrary.ru/download/elibrary_49415977_56758859.pdf#page=36 (дата обращения: 24.05.2024).

20. Сааков В. В. и др. вопросы конфиденциальности и безопасности nft // современная наука и молодые учёные 3. – 2022. – С. 52. URL: https://naukaip.ru/wp-content/uploads/2022/06/MK-1440.pdf#page=52 (дата обращения: 24.05.2024).

21. Сафиуллин Марат Рашитович, Абдукаева Алия Айдаровна, Ельшин Леонид Алексеевич Методические подходы к прогнозированию динамики курса криптовалют с применением инструментов стохастического анализа (на примере биткоина) // Финансы: теория и практика. 2018. №4. URL: https://cyberleninka.ru/article/n/metodicheskie-podhody-k-prognozirovaniyu-dinamiki-kursa-kriptovalyut-s-primeneniem-instrumentov-stohasticheskogo-analiza-na-primere (дата обращения: 24.05.2024).

22. Сидоренко Валентина Геннадьевна, Сафронов Антон Игоревич применение генетических алгоритмов при решении задач планирования перевозочного процесса городской рельсовой транспортной системы // Автоматика на транспорте. 2023. №1. URL: https://cyberleninka.ru/article/n/primenenie-geneticheskih-algoritmov-pri-reshenii-zadach-planirovaniya-perevozochnogo-protsessa-gorodskoy-relsovoy-transportnoy (дата обращения: 24.05.2024).

23. Слепнёв С. В., Коледина К. Ф. прогнозирование кинетики сложных процессов люминесценции на python // Computational Mathematics and Information Technologies. 2023. №1. URL: https://cyberleninka.ru/article/n/prognozirovanie-

kinetiki-slozhnyh-protsessov-lyuminestsentsii-na-python (дата обращения: 24.05.2024).

24. Тулегенов А.А. генетический криптоалгоритм // Перспективы развития информационных технологий. 2016. №29. URL: https://cyberleninka.ru/article/n/geneticheskiy-kriptoalgoritm (дата обращения: 24.05.2024).

25. Фельдман Елена Васильевна, Ручай Алексей Николаевич, Матвеева Вероника Константиновна, Самсонова Валерия Дмитриевна модель выявления аномальных транзакций биткоинов на основе машинного обучения // Челябинский физико-математический журнал. 2021. №1. URL: https://cyberleninka.ru/article/n/model-vyyavleniya-anomalnyh-tranzaktsiy-bitkoinov-na-osnove-mashinnogo-obucheniya (дата обращения: 24.05.2024).

26. Цыганков В.А., Шабалина О.А., Катаев А.В. использование генетических алгоритмов для повышения скорости обучения нейронных сетей // ИВД. 2023. №2 (98). URL: https://cyberleninka.ru/article/n/ispolzovanie-geneticheskih-algoritmov-dlya-povysheniya-skorosti-obucheniya-neyronnyh-setey (дата обращения: 24.05.2024).

27. Чаплыгин Владимир Германович, Дельцова Татьяна Александровна Корреляционно-регрессионный анализ взаимосвязи признаков криптовалюты биткоин // Вестник Балтийского федерального университета им. И. Канта. Серия: Гуманитарные и общественные науки. 2018. №2. URL: https://cyberleninka.ru/article/n/korrelyatsionno-regressionnyy-analiz-vzaimosvyazi-priznakov-kriptovalyuty-bitkoin (дата обращения: 24.05.2024).

28. Bitcoin Price Dataset (2014-2023) URL: https://finance.yahoo.com/quote/BTC-USD/ (дата обращения: 24.05.2024).

ПРИЛОЖЕНИЕ А

(обязательное)

Использование генетических алгоритмов в интеллектуальном анализе данных

Листинг программы

Файл train_model.py:

```
import os
import pandas as pd
import numpy as np
import math
import datetime as dt

# For Evaluation we will use these libraries

from sklearn.metrics import mean_squared_error,
mean_absolute_error, explained_variance_score, r2_score
from sklearn.metrics import mean_poisson_deviance,
mean_gamma_deviance, accuracy_score
from sklearn.preprocessing import MinMaxScaler

# For model building we will use these libraries

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.layers import LSTM

# For Plotting we will use these libraries

import matplotlib.pyplot as plt
from itertools import cycle
import plotly.graph_objects as go
import plotly.express as px
from plotly.subplots import make_subplots

"""# 3. Загрузка Датасета

#  [We can use this link to download bitcoin dataset from yahoo
finance](https://finance.yahoo.com/quote/BTC-USD/history?p=BTC-
USD)

- or can use a script to download the latest data by calling
yfinance api.(Check Data Folder)
"""

# Load our dataset
# Note it should be in same dir

df_bitcoin = pd.read_csv('BTC-USD.csv')

print('Total number of days present in the dataset: ',
df_bitcoin.shape[0])
print('Total number of fields present in the dataset: ',
df_bitcoin.shape[1])

df_bitcoin.shape
```

```
df_bitcoin.head()

df_bitcoin.tail()

df_bitcoin.info()

df_bitcoin.describe()

"""# 3.Проверка нулевых элементов"""

print('Null Values:', df_bitcoin.isnull().values.sum())

print('NA values:', df_bitcoin.isnull().values.any())

# If dataset had null values we can use this code to drop all
the null values present in the dataset

# df_bitcoin = df_bitcoin.dropna()
# print('Null Values:', df_bitcoin.isnull().values.sum())
# print('NA values:', df_bitcoin.isnull().values.any())

# Final shape of the dataset after dealing with null values

df_bitcoin.shape

"""# 4. EDA (Исследовательский анализ данных)"""

# Printing the start date and End date of the dataset

start_date = df_bitcoin.iloc[0][0]
end_date = df_bitcoin.iloc[-1][0]

print('Starting Date', start_date)
print('Ending Date', end_date)

"""#### StockPrice Analysis from Start

# Analysis of year 2014
"""

df_bitcoin['Date'] = pd.to_datetime(df_bitcoin['Date'],
format='%Y-%m-%d')

df_2014 = df_bitcoin.loc[(df_bitcoin['Date'] >= '2014-09-17')
                         & (df_bitcoin['Date'] < '2014-12-31')]

df_2014.drop(df_2014[['Adj Close', 'Volume']], axis=1)

df_monthly_2014 =
df_2014.groupby(df_2014['Date'].dt.strftime('%B'))[['Open',
'Close']].mean()
month_order = ['January', 'February', 'March', 'April', 'May',
'June', 'July', 'August',
```

```
                'September', 'October', 'November', 'December']
df_monthly_2014 = df_monthly_2014.reindex(month_order, axis=0)
df_monthly_2014

""" Note that we only have few months in 2014 so the rest of the
months are **NaN**"""

fig = go.Figure()

fig.add_trace(go.Bar(
    x=df_monthly_2014.index,
    y=df_monthly_2014['Open'],
    name='Bitcoin Open Price',
    marker_color='crimson'
))
fig.add_trace(go.Bar(
    x=df_monthly_2014.index,
    y=df_monthly_2014['Close'],
    name='Bitcoin Close Price',
    marker_color='lightsalmon'
))

fig.update_layout(barmode='group', xaxis_tickangle=-45,
                  title='Monthwise comparison between Bitcoin
open and close price')
fig.show()

"""Note that we only have few months in 2014 so the rest of the
months are not plotted since we do not have the data"""

df_2014.groupby(df_2014['Date'].dt.strftime('%B'))['Low'].min()
df_monthly_high_2014 =
df_2014.groupby(df_bitcoin['Date'].dt.strftime('%B'))['High'].ma
x()
df_monthly_high_2014 = df_monthly_high_2014.reindex(month_order,
axis=0)

df_monthly_low_2014 =
df_2014.groupby(df_2014['Date'].dt.strftime('%B'))['Low'].min()
df_monthly_low_2014 = df_monthly_low_2014.reindex(month_order,
axis=0)

fig = go.Figure()
fig.add_trace(go.Bar(
    x=df_monthly_high_2014.index,
    y=df_monthly_high_2014,
    name='Bitcoin High Price',
    marker_color='rgb(0, 153, 204)'
))
fig.add_trace(go.Bar(
    x=df_monthly_low_2014.index,
    y=df_monthly_low_2014,
    name='Bitcoin Low Price',
```

```
    marker_color='rgb(255, 128, 0)'
))

fig.update_layout(barmode='group',
                  title='Monthwise High and Low Bitcoin price')
fig.show()

"""Note that we only have few months in 2014 so the rest of the
months are not plotted since we do not have the data"""

names = cycle(['Bitcoin Open Price', 'Bitcoin Close Price',
'Bitcoin High Price', 'Bitcoin Low Price'])

fig = px.line(df_2014, x=df_2014.Date, y=[df_2014['Open'],
df_2014['Close'],
                                          df_2014['High'],
df_2014['Low']],
              labels={'Date': 'Date', 'value': 'Bitcoin value'})
fig.update_layout(title_text='Bitcoin analysis chart',
font_size=15, font_color='black', legend_title_text='Bitcoin
Parameters')
fig.for_each_trace(lambda t: t.update(name=next(names)))
fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)

fig.show()

"""Note that we only have few months in 2014 so the rest of the
months are not plotted since we do not have the data

# Analysis of Year 2015
"""

df_bitcoin['Date'] = pd.to_datetime(df_bitcoin['Date'],
format='%Y-%m-%d')

df_2015 = df_bitcoin.loc[(df_bitcoin['Date'] >= '2015-01-01')
                         & (df_bitcoin['Date'] < '2016-01-01')]

df_2015.drop(df_2015[['Adj Close', 'Volume']], axis=1)

df_monthly_2015 =
df_2015.groupby(df_2015['Date'].dt.strftime('%B'))[['Open',
'Close']].mean()
df_monthly_2015 = df_monthly_2015.reindex(month_order, axis=0)
df_monthly_2015

fig = go.Figure()

fig.add_trace(go.Bar(
    x=df_monthly_2015.index,
    y=df_monthly_2015['Open'],
    name='Stock Open Price',
```

```
    marker_color='crimson'
))
fig.add_trace(go.Bar(
    x=df_monthly_2015.index,
    y=df_monthly_2015['Close'],
    name='Stock Close Price',
    marker_color='lightsalmon'
))

fig.update_layout(barmode='group', xaxis_tickangle=-45,
                  title='Monthwise comparison between Stock open
and close price')
fig.show()

df_2015.groupby(df_2015['Date'].dt.strftime('%B'))['Low'].min()
df_monthly_high_2015 =
df_2015.groupby(df_bitcoin['Date'].dt.strftime('%B'))['High'].ma
x()
df_monthly_high_2015 = df_monthly_high_2015.reindex(month_order,
axis=0)

df_monthly_low_2015 =
df_2015.groupby(df_2015['Date'].dt.strftime('%B'))['Low'].min()
df_monthly_low_2015 = df_monthly_low_2015.reindex(month_order,
axis=0)

fig = go.Figure()
fig.add_trace(go.Bar(
    x=df_monthly_high_2015.index,
    y=df_monthly_high_2015,
    name='Stock High Price',
    marker_color='rgb(0, 153, 204)'
))
fig.add_trace(go.Bar(
    x=df_monthly_low_2015.index,
    y=df_monthly_low_2015,
    name='Stock Low Price',
    marker_color='rgb(255, 128, 0)'
))

fig.update_layout(barmode='group',
                  title='Monthwise High and Low stock price')
fig.show()

names = cycle(['Stock Open Price', 'Stock Close Price', 'Stock
High Price', 'Stock Low Price'])

fig = px.line(df_2015, x=df_2015.Date, y=[df_2015['Open'],
df_2015['Close'],
                                          df_2015['High'],
df_2015['Low']],
              labels={'Date': 'Date', 'value': 'Stock value'})
```

```
fig.update_layout(title_text='Stock analysis chart',
font_size=15, font_color='black', legend_title_text='Stock
Parameters')
fig.for_each_trace(lambda t: t.update(name=next(names)))
fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)

fig.show()

"""# Analysis of Year 2016"""

df_bitcoin['Date'] = pd.to_datetime(df_bitcoin['Date'],
format='%Y-%m-%d')

df_2016 = df_bitcoin.loc[(df_bitcoin['Date'] >= '2016-01-01')
                         & (df_bitcoin['Date'] < '2017-01-01')]

df_2016.drop(df_2016[['Adj Close', 'Volume']], axis=1)

df_monthly_2016 =
df_2016.groupby(df_2016['Date'].dt.strftime('%B'))[['Open',
'Close']].mean()
df_monthly_2016 = df_monthly_2016.reindex(month_order, axis=0)
df_monthly_2016

fig = go.Figure()

fig.add_trace(go.Bar(
    x=df_monthly_2016.index,
    y=df_monthly_2016['Open'],
    name='Stock Open Price',
    marker_color='crimson'
))
fig.add_trace(go.Bar(
    x=df_monthly_2016.index,
    y=df_monthly_2016['Close'],
    name='Stock Close Price',
    marker_color='lightsalmon'
))

fig.update_layout(barmode='group', xaxis_tickangle=-45,
                  title='Monthwise comparison between Stock open
and close price')
fig.show()

df_2016.groupby(df_2016['Date'].dt.strftime('%B'))['Low'].min()
df_monthly_high_2016 =
df_2016.groupby(df_bitcoin['Date'].dt.strftime('%B'))['High'].ma
x()
df_monthly_high_2016 = df_monthly_high_2016.reindex(month_order,
axis=0)
```

```
df_monthly_low_2016 =
df_2016.groupby(df_2016['Date'].dt.strftime('%B'))['Low'].min()
df_monthly_low_2016 = df_monthly_low_2016.reindex(month_order,
axis=0)

fig = go.Figure()
fig.add_trace(go.Bar(
    x=df_monthly_high_2016.index,
    y=df_monthly_high_2016,
    name='Stock High Price',
    marker_color='rgb(0, 153, 204)'
))
fig.add_trace(go.Bar(
    x=df_monthly_low_2016.index,
    y=df_monthly_low_2016,
    name='Stock Low Price',
    marker_color='rgb(255, 128, 0)'
))

fig.update_layout(barmode='group',
                  title='Monthwise High and Low stock price')
fig.show()

names = cycle(['Stock Open Price', 'Stock Close Price', 'Stock
High Price', 'Stock Low Price'])

fig = px.line(df_2016, x=df_2016.Date, y=[df_2016['Open'],
df_2016['Close'],
                                          df_2016['High'],
df_2016['Low']],
              labels={'Date': 'Date', 'value': 'Stock value'})
fig.update_layout(title_text='Stock analysis chart',
font_size=15, font_color='black', legend_title_text='Stock
Parameters')
fig.for_each_trace(lambda t: t.update(name=next(names)))
fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)

fig.show()

"""# Analysis of Year 2017"""

df_bitcoin['Date'] = pd.to_datetime(df_bitcoin['Date'],
format='%Y-%m-%d')

df_2017 = df_bitcoin.loc[(df_bitcoin['Date'] >= '2017-01-01')
                         & (df_bitcoin['Date'] < '2018-01-01')]

df_2017.drop(df_2017[['Adj Close', 'Volume']], axis=1)

df_monthly_2017 =
df_2017.groupby(df_2017['Date'].dt.strftime('%B'))[['Open',
'Close']].mean()
```

```
df_monthly_2017 = df_monthly_2017.reindex(month_order, axis=0)
df_monthly_2017

fig = go.Figure()

fig.add_trace(go.Bar(
    x=df_monthly_2017.index,
    y=df_monthly_2017['Open'],
    name='Stock Open Price',
    marker_color='crimson'
))
fig.add_trace(go.Bar(
    x=df_monthly_2017.index,
    y=df_monthly_2017['Close'],
    name='Stock Close Price',
    marker_color='lightsalmon'
))

fig.update_layout(barmode='group', xaxis_tickangle=-45,
                  title='Monthwise comparison between Stock open
and close price')
fig.show()

df_2017.groupby(df_2017['Date'].dt.strftime('%B'))['Low'].min()
df_monthly_high_2017 =
df_2017.groupby(df_bitcoin['Date'].dt.strftime('%B'))['High'].ma
x()
df_monthly_high_2017 = df_monthly_high_2017.reindex(month_order,
axis=0)

df_monthly_low_2017 =
df_2017.groupby(df_2017['Date'].dt.strftime('%B'))['Low'].min()
df_monthly_low_2017 = df_monthly_low_2017.reindex(month_order,
axis=0)

fig = go.Figure()
fig.add_trace(go.Bar(
    x=df_monthly_high_2017.index,
    y=df_monthly_high_2017,
    name='Stock High Price',
    marker_color='rgb(0, 153, 204)'
))
fig.add_trace(go.Bar(
    x=df_monthly_low_2017.index,
    y=df_monthly_low_2017,
    name='Stock Low Price',
    marker_color='rgb(255, 128, 0)'
))

fig.update_layout(barmode='group',
                  title='Monthwise High and Low stock price')
fig.show()
```

```
names = cycle(['Stock Open Price', 'Stock Close Price', 'Stock
High Price', 'Stock Low Price'])

fig = px.line(df_2017, x=df_2017.Date, y=[df_2017['Open'],
df_2017['Close'],
                                          df_2017['High'],
df_2017['Low']],
              labels={'Date': 'Date', 'value': 'Stock value'})
fig.update_layout(title_text='Stock analysis chart',
font_size=15, font_color='black', legend_title_text='Stock
Parameters')
fig.for_each_trace(lambda t: t.update(name=next(names)))
fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)

fig.show()

"""# Analysis of Year 2018"""

df_bitcoin['Date'] = pd.to_datetime(df_bitcoin['Date'],
format='%Y-%m-%d')

df_2018 = df_bitcoin.loc[(df_bitcoin['Date'] >= '2018-01-01')
                         & (df_bitcoin['Date'] < '2019-01-01')]

df_2018.drop(df_2018[['Adj Close', 'Volume']], axis=1)

df_monthly_2018 =
df_2018.groupby(df_2018['Date'].dt.strftime('%B'))[['Open',
'Close']].mean()
df_monthly_2018 = df_monthly_2018.reindex(month_order, axis=0)
df_monthly_2018

fig = go.Figure()

fig.add_trace(go.Bar(
    x=df_monthly_2018.index,
    y=df_monthly_2018['Open'],
    name='Stock Open Price',
    marker_color='crimson'
))
fig.add_trace(go.Bar(
    x=df_monthly_2018.index,
    y=df_monthly_2018['Close'],
    name='Stock Close Price',
    marker_color='lightsalmon'
))

fig.update_layout(barmode='group', xaxis_tickangle=-45,
                  title='Monthwise comparison between Stock open
and close price')
fig.show()
```

```
df_2018.groupby(df_2018['Date'].dt.strftime('%B'))['Low'].min()
df_monthly_high_2018 =
df_2018.groupby(df_bitcoin['Date'].dt.strftime('%B'))['High'].ma
x()
df_monthly_high_2018 = df_monthly_high_2018.reindex(month_order,
axis=0)

df_monthly_low_2018 =
df_2018.groupby(df_2018['Date'].dt.strftime('%B'))['Low'].min()
df_monthly_low_2018 = df_monthly_low_2018.reindex(month_order,
axis=0)

fig = go.Figure()
fig.add_trace(go.Bar(
    x=df_monthly_high_2018.index,
    y=df_monthly_high_2018,
    name='Stock High Price',
    marker_color='rgb(0, 153, 204)'
))
fig.add_trace(go.Bar(
    x=df_monthly_low_2018.index,
    y=df_monthly_low_2018,
    name='Stock Low Price',
    marker_color='rgb(255, 128, 0)'
))

fig.update_layout(barmode='group',
                  title='Monthwise High and Low stock price')
fig.show()

names = cycle(['Stock Open Price', 'Stock Close Price', 'Stock
High Price', 'Stock Low Price'])

fig = px.line(df_2018, x=df_2018.Date, y=[df_2018['Open'],
df_2018['Close'],
                                          df_2018['High'],
df_2018['Low']],
              labels={'Date': 'Date', 'value': 'Stock value'})
fig.update_layout(title_text='Stock analysis chart',
font_size=15, font_color='black', legend_title_text='Stock
Parameters')
fig.for_each_trace(lambda t: t.update(name=next(names)))
fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)

fig.show()

"""# Analysis of Year 2019"""

df_bitcoin['Date'] = pd.to_datetime(df_bitcoin['Date'],
format='%Y-%m-%d')

df_2019 = df_bitcoin.loc[(df_bitcoin['Date'] >= '2019-01-01')
```

```
                          & (df_bitcoin['Date'] < '2020-01-01')]

df_2019.drop(df_2019[['Adj Close', 'Volume']], axis=1)

df_monthly_2019 =
df_2019.groupby(df_2019['Date'].dt.strftime('%B'))[['Open',
'Close']].mean()
df_monthly_2019 = df_monthly_2019.reindex(month_order, axis=0)
df_monthly_2019

fig = go.Figure()

fig.add_trace(go.Bar(
    x=df_monthly_2019.index,
    y=df_monthly_2019['Open'],
    name='Stock Open Price',
    marker_color='crimson'
))
fig.add_trace(go.Bar(
    x=df_monthly_2019.index,
    y=df_monthly_2019['Close'],
    name='Stock Close Price',
    marker_color='lightsalmon'
))

fig.update_layout(barmode='group', xaxis_tickangle=-45,
                  title='Monthwise comparison between Stock open
and close price')
fig.show()

df_2019.groupby(df_2019['Date'].dt.strftime('%B'))['Low'].min()
df_monthly_high_2019 =
df_2019.groupby(df_bitcoin['Date'].dt.strftime('%B'))['High'].ma
x()
df_monthly_high_2019 = df_monthly_high_2019.reindex(month_order,
axis=0)

df_monthly_low_2019 =
df_2019.groupby(df_2019['Date'].dt.strftime('%B'))['Low'].min()
df_monthly_low_2019 = df_monthly_low_2019.reindex(month_order,
axis=0)

fig = go.Figure()
fig.add_trace(go.Bar(
    x=df_monthly_high_2019.index,
    y=df_monthly_high_2019,
    name='Stock High Price',
    marker_color='rgb(0, 153, 204)'
))
fig.add_trace(go.Bar(
    x=df_monthly_low_2019.index,
    y=df_monthly_low_2019,
    name='Stock Low Price',
```

```
    marker_color='rgb(255, 128, 0)'
))

fig.update_layout(barmode='group',
                  title='Monthwise High and Low stock price')
fig.show()

names = cycle(['Stock Open Price', 'Stock Close Price', 'Stock
High Price', 'Stock Low Price'])

fig = px.line(df_2019, x=df_2019.Date, y=[df_2019['Open'],
df_2019['Close'],
                                          df_2019['High'],
df_2019['Low']],
              labels={'Date': 'Date', 'value': 'Stock value'})
fig.update_layout(title_text='Stock analysis chart',
font_size=15, font_color='black', legend_title_text='Stock
Parameters')
fig.for_each_trace(lambda t: t.update(name=next(names)))
fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)

fig.show()

"""# Analysis of Year 2020"""

df_bitcoin['Date'] = pd.to_datetime(df_bitcoin['Date'],
format='%Y-%m-%d')

df_2020 = df_bitcoin.loc[(df_bitcoin['Date'] >= '2020-01-01')
                         & (df_bitcoin['Date'] < '2021-01-01')]

df_2020.drop(df_2020[['Adj Close', 'Volume']], axis=1)

df_monthly_2020 =
df_2020.groupby(df_2020['Date'].dt.strftime('%B'))[['Open',
'Close']].mean()
df_monthly_2020 = df_monthly_2020.reindex(month_order, axis=0)
df_monthly_2020

fig = go.Figure()

fig.add_trace(go.Bar(
    x=df_monthly_2020.index,
    y=df_monthly_2020['Open'],
    name='Stock Open Price',
    marker_color='crimson'
))
fig.add_trace(go.Bar(
    x=df_monthly_2020.index,
    y=df_monthly_2020['Close'],
    name='Stock Close Price',
    marker_color='lightsalmon'
```

```
))

fig.update_layout(barmode='group', xaxis_tickangle=-45,
                  title='Monthwise comparison between Stock open
and close price')
fig.show()

df_2020.groupby(df_2020['Date'].dt.strftime('%B'))['Low'].min()
df_monthly_high_2020 =
df_2020.groupby(df_bitcoin['Date'].dt.strftime('%B'))['High'].ma
x()
df_monthly_high_2020 = df_monthly_high_2020.reindex(month_order,
axis=0)

df_monthly_low_2020 =
df_2020.groupby(df_2020['Date'].dt.strftime('%B'))['Low'].min()
df_monthly_low_2020 = df_monthly_low_2020.reindex(month_order,
axis=0)

fig = go.Figure()
fig.add_trace(go.Bar(
    x=df_monthly_high_2020.index,
    y=df_monthly_high_2020,
    name='Stock High Price',
    marker_color='rgb(0, 153, 204)'
))
fig.add_trace(go.Bar(
    x=df_monthly_low_2020.index,
    y=df_monthly_low_2020,
    name='Stock Low Price',
    marker_color='rgb(255, 128, 0)'
))

fig.update_layout(barmode='group',
                  title='Monthwise High and Low stock price')
fig.show()

names = cycle(['Stock Open Price', 'Stock Close Price', 'Stock
High Price', 'Stock Low Price'])

fig = px.line(df_2020, x=df_2020.Date, y=[df_2020['Open'],
df_2020['Close'],
                                          df_2020['High'],
df_2020['Low']],
              labels={'Date': 'Date', 'value': 'Stock value'})
fig.update_layout(title_text='Stock analysis chart',
font_size=15, font_color='black', legend_title_text='Stock
Parameters')
fig.for_each_trace(lambda t: t.update(name=next(names)))
fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)

fig.show()
```

```
"""# Analysis of Year 2021"""

df_bitcoin['Date'] = pd.to_datetime(df_bitcoin['Date'],
format='%Y-%m-%d')

df_2021 = df_bitcoin.loc[(df_bitcoin['Date'] >= '2021-01-01')
                         & (df_bitcoin['Date'] < '2021-12-31')]

df_2021.drop(df_2021[['Adj Close', 'Volume']], axis=1)

df_monthly_2021 =
df_2021.groupby(df_2021['Date'].dt.strftime('%B'))[['Open',
'Close']].mean()
df_monthly_2021 = df_monthly_2021.reindex(month_order, axis=0)
df_monthly_2021

"""#### Since we had data till 24-08-2021 in Months after August
its showing NaN"""

fig = go.Figure()

fig.add_trace(go.Bar(
    x=df_monthly_2021.index,
    y=df_monthly_2021['Open'],
    name='Stock Open Price',
    marker_color='crimson'
))
fig.add_trace(go.Bar(
    x=df_monthly_2021.index,
    y=df_monthly_2021['Close'],
    name='Stock Close Price',
    marker_color='lightsalmon'
))

fig.update_layout(barmode='group', xaxis_tickangle=-45,
                  title='Monthwise comparison between Stock open
and close price')
fig.show()

df_2021.groupby(df_2021['Date'].dt.strftime('%B'))['Low'].min()
df_monthly_high_2021 =
df_2021.groupby(df_bitcoin['Date'].dt.strftime('%B'))['High'].ma
x()
df_monthly_high_2021 = df_monthly_high_2021.reindex(month_order,
axis=0)

df_monthly_low_2021 =
df_2021.groupby(df_2021['Date'].dt.strftime('%B'))['Low'].min()
df_monthly_low_2021 = df_monthly_low_2021.reindex(month_order,
axis=0)

fig = go.Figure()
```

```
fig.add_trace(go.Bar(
    x=df_monthly_high_2021.index,
    y=df_monthly_high_2021,
    name='Stock High Price',
    marker_color='rgb(0, 153, 204)'
))
fig.add_trace(go.Bar(
    x=df_monthly_low_2021.index,
    y=df_monthly_low_2021,
    name='Stock Low Price',
    marker_color='rgb(255, 128, 0)'
))

fig.update_layout(barmode='group',
                  title='Monthwise High and Low stock price')
fig.show()

names = cycle(['Stock Open Price', 'Stock Close Price', 'Stock
High Price', 'Stock Low Price'])

fig = px.line(df_2021, x=df_2021.Date, y=[df_2021['Open'],
df_2021['Close'],
                                          df_2021['High'],
df_2021['Low']],
              labels={'Date': 'Date', 'value': 'Stock value'})
fig.update_layout(title_text='Stock analysis chart',
font_size=15, font_color='black', legend_title_text='Stock
Parameters')
fig.for_each_trace(lambda t: t.update(name=next(names)))
fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)

fig.show()

"""# Analysis of Year 2022"""

df_bitcoin['Date'] = pd.to_datetime(df_bitcoin['Date'],
format='%Y-%m-%d')

df_2022 = df_bitcoin.loc[(df_bitcoin['Date'] >= '2022-01-01')
                         & (df_bitcoin['Date'] < '2022-02-19')]

df_2022.drop(df_2022[['Adj Close', 'Volume']], axis=1)

df_monthly_2022 =
df_2022.groupby(df_2022['Date'].dt.strftime('%B'))[['Open',
'Close']].mean()
df_monthly_2022 = df_monthly_2022.reindex(month_order, axis=0)
df_monthly_2022

fig = go.Figure()

fig.add_trace(go.Bar(
```

```
    x=df_monthly_2022.index,
    y=df_monthly_2022['Open'],
    name='Stock Open Price',
    marker_color='crimson'
))
fig.add_trace(go.Bar(
    x=df_monthly_2022.index,
    y=df_monthly_2022['Close'],
    name='Stock Close Price',
    marker_color='lightsalmon'
))

fig.update_layout(barmode='group', xaxis_tickangle=-45,
                  title='Monthwise comparison between Stock open
and close price')
fig.show()

df_2022.groupby(df_2022['Date'].dt.strftime('%B'))['Low'].min()
df_monthly_high_2022 =
df_2022.groupby(df_bitcoin['Date'].dt.strftime('%B'))['High'].ma
x()
df_monthly_high_2022 = df_monthly_high_2022.reindex(month_order,
axis=0)

df_monthly_low_2022 =
df_2022.groupby(df_2022['Date'].dt.strftime('%B'))['Low'].min()
df_monthly_low_2022 = df_monthly_low_2022.reindex(month_order,
axis=0)

fig = go.Figure()
fig.add_trace(go.Bar(
    x=df_monthly_high_2022.index,
    y=df_monthly_high_2022,
    name='Stock High Price',
    marker_color='rgb(0, 153, 204)'
))
fig.add_trace(go.Bar(
    x=df_monthly_low_2022.index,
    y=df_monthly_low_2022,
    name='Stock Low Price',
    marker_color='rgb(255, 128, 0)'
))

fig.update_layout(barmode='group',
                  title='Monthwise High and Low stock price')
fig.show()

names = cycle(['Stock Open Price', 'Stock Close Price', 'Stock
High Price', 'Stock Low Price'])

fig = px.line(df_2022, x=df_2022.Date, y=[df_2022['Open'],
df_2022['Close'],
```

```
                                            df_2022['High'],
df_2022['Low']],
              labels={'Date': 'Date', 'value': 'Stock value'})
fig.update_layout(title_text='Stock analysis chart',
font_size=15, font_color='black', legend_title_text='Stock
Parameters')
fig.for_each_trace(lambda t: t.update(name=next(names)))
fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)

fig.show()

"""# Overall Analysis from 2014-2022"""

df_bitcoin['Date'] = pd.to_datetime(df_bitcoin['Date'],
format='%Y-%m-%d')

df_overall = df_bitcoin.loc[(df_bitcoin['Date'] >= '2014-09-17')
                            & (df_bitcoin['Date'] <= '2022-02-
19')]

df_overall.drop(df_overall[['Adj Close', 'Volume']], axis=1)

df_monthly_overall =
df_overall.groupby(df_overall['Date'].dt.strftime('%B'))[['Open'
, 'Close']].mean()
df_monthly_overall = df_monthly_overall.reindex(month_order,
axis=0)
df_monthly_overall

names = cycle(['Stock Open Price', 'Stock Close Price', 'Stock
High Price', 'Stock Low Price'])

fig = px.line(df_overall, x=df_overall.Date,
y=[df_overall['Open'], df_overall['Close'],

df_overall['High'], df_overall['Low']],
              labels={'Date': 'Date', 'value': 'Stock value'})
fig.update_layout(title_text='Stock analysis chart',
font_size=15, font_color='black', legend_title_text='Stock
Parameters')
fig.for_each_trace(lambda t: t.update(name=next(names)))
fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)

fig.show()

"""# 5. Делаем LSTM Model

* ## Первым шагом является подготовка данных для обучения и
тестирования
```

```
* ## Здесь мы рассматриваем данные только за 1 год для целей
обучения

* ## Поскольку цена биткоина резко колебалась с 200 долларов в
2014 году до 15000 долларов в 2018 году и до 3000 долларов в
2019 году (эти значения пропорциональны), мы рассмотрим только 1
год, чтобы избежать такого рода колебаний в данных.

* ## Поскольку мы хотим предсказать цену закрытия Биткоина, мы
просто рассматриваем дату закрытия
"""

# Lets First Take all the Close Price
df_close = df_bitcoin[['Date', 'Close']]
print("Shape of close dataframe:", df_close.shape)

fig = px.line(df_close, x=df_close.Date, y=df_close.Close,
labels={'date': 'Date', 'close': 'Close Stock'})
fig.update_traces(marker_line_width=2, opacity=0.8,
marker_line_color='orange')
fig.update_layout(title_text='Весь период времени закрытия цены
биткоина 2014-2022', plot_bgcolor='white',
                  font_size=15, font_color='black')
fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)
fig.show()

"""### Теперь мы возьмем данные всего за 1 год"""

df_close = df_close[df_close['Date'] > '2021-02-19']
df_close_stock = df_close.copy()
print("Total data for prediction: ", df_close.shape[0])

df_close

fig = px.line(df_close, x=df_close.Date, y=df_close.Close,
labels={'date': 'Date', 'close': 'Close Stock'})
fig.update_traces(marker_line_width=2, opacity=0.8,
marker_line_color='orange')
fig.update_layout(title_text='Рассматриваемый период для
прогнозирования цены закрытия биткойна',
                  plot_bgcolor='white', font_size=15,
font_color='black')
fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)
fig.show()

"""* ### Нормализация данных
- Нормализация - это метод, часто применяемый при подготовке
данных для машинного обучения. Цель нормализации - изменить
значения числовых столбцов в наборе данных, чтобы использовать
общую шкалу, не искажая различия в диапазонах значений и не
теряя информацию.
```

```
- MinMaxScaler. Для каждого значения в объекте MinMaxScaler
вычитает минимальное значение в объекте, а затем делит на
диапазон. Диапазон - это разница между исходным максимумом и
исходным минимумом. MinMaxScaler сохраняет форму исходного
дистрибутива.
"""

# deleting date column and normalizing using MinMax Scaler

del df_close['Date']
scaler = MinMaxScaler(feature_range=(0, 1))
df_close = scaler.fit_transform(np.array(df_close).reshape(-1,
1))
print(df_close.shape)

"""* ### Разделение данных на обучающий набор и тестовый
набор"""

# we keep the training set as 60% and 40% testing set

train_size = int(len(df_close) * 0.60)
test_size = len(df_close) - train_size
train_data, test_data = df_close[0:train_size, :],
df_close[train_size:len(df_close), :1]
print("train_data: ", train_data.shape)
print("test_data: ", test_data.shape)

"""* ### Теперь мы преобразуем цену закрытия на основе
требований прогнозирования с помощью анализа временных рядов,
здесь мы возьмем 15"""

# convert an array of values into a dataset matrix

def create_dataset(dataset, time_step=1):
    dataX, dataY = [], []
    for i in range(len(dataset) - time_step - 1):
        a = dataset[i:(i + time_step), 0]  ###i=0, 0,1,2,3-----
99   100
        dataX.append(a)
        dataY.append(dataset[i + time_step, 0])
    return np.array(dataX), np.array(dataY)

time_step = 15
X_train, y_train = create_dataset(train_data, time_step)
X_test, y_test = create_dataset(test_data, time_step)

print("X_train: ", X_train.shape)
print("y_train: ", y_train.shape)
print("X_test: ", X_test.shape)
print("y_test", y_test.shape)

# reshape input to be [samples, time steps, features] which is
required for LSTM
```

```
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], 1)
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], 1)

print("X_train: ", X_train.shape)
print("X_test: ", X_test.shape)

"""* # Построение реальной модели"""

model = Sequential()

model.add(LSTM(10, input_shape=(None, 1), activation="relu"))

model.add(Dense(1))

model.compile(loss="mean_squared_error", optimizer="adam")

history = model.fit(X_train, y_train, validation_data=(X_test,
y_test), epochs=200, batch_size=32, verbose=1)

"""- ### Plotting Loss vs Validation loss"""

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(len(loss))

plt.plot(epochs, loss, 'r', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend(loc=0)
plt.figure()

plt.show()

### Lets Do the prediction and check performance metrics
train_predict = model.predict(X_train)
test_predict = model.predict(X_test)
train_predict.shape, test_predict.shape

"""- # Оценка модели"""

# Transform back to original form

train_predict = scaler.inverse_transform(train_predict)
test_predict = scaler.inverse_transform(test_predict)
original_ytrain = scaler.inverse_transform(y_train.reshape(-1,
1))
original_ytest = scaler.inverse_transform(y_test.reshape(-1, 1))

"""- ## Оценочные метрики RMSE, MSE и MAE"""

# Evaluation metrics RMSE and MAE
```

```
print("Train data RMSE: ",
math.sqrt(mean_squared_error(original_ytrain, train_predict)))
print("Train data MSE: ", mean_squared_error(original_ytrain,
train_predict))
print("Train data MAE: ", mean_absolute_error(original_ytrain,
train_predict))
print("-------------------------------------------------------
-----------------------------")
print("Test data RMSE: ",
math.sqrt(mean_squared_error(original_ytest, test_predict)))
print("Test data MSE: ", mean_squared_error(original_ytest,
test_predict))
print("Test data MAE: ", mean_absolute_error(original_ytest,
test_predict))

"""- ## Оценка регрессии дисперсии"""

print("Train data explained variance regression score:",
      explained_variance_score(original_ytrain, train_predict))
print("Test data explained variance regression score:",
      explained_variance_score(original_ytest, test_predict))

"""- ## R квадратный балл для регрессии"""

print("Train data R2 score:", r2_score(original_ytrain,
train_predict))
print("Test data R2 score:", r2_score(original_ytest,
test_predict))

"""- ## Потеря регрессии Означает потерю регрессии гамма-
отклонения (MGD) и среднюю потерю регрессии пуассоновского
отклонения (MPD)."""

print("Train data MGD: ", mean_gamma_deviance(original_ytrain,
train_predict))
print("Test data MGD: ", mean_gamma_deviance(original_ytest,
test_predict))
print("-------------------------------------------------------
-------------")
print("Train data MPD: ", mean_poisson_deviance(original_ytrain,
train_predict))
print("Test data MPD: ", mean_poisson_deviance(original_ytest,
test_predict))

"""- # Сравнение первоначальной цены закрытия сделки с биткоином
и прогнозируемой цены закрытия сделки"""

# shift train predictions for plotting

look_back = time_step
train_predict_plot = np.empty_like(df_close)
train_predict_plot[:, :] = np.nan
```

```
train_predict_plot[look_back:len(train_predict) + look_back, :]
= train_predict
print("Train predicted data: ", train_predict_plot.shape)

# shift test predictions for plotting
test_predict_plot = np.empty_like(df_close)
test_predict_plot[:, :] = np.nan
test_predict_plot[len(train_predict) + (look_back * 2) +
1:len(df_close) - 1, :] = test_predict
print("Test predicted data: ", test_predict_plot.shape)

names = cycle(['Original close price', 'Train predicted close
price', 'Test predicted close price'])

df_plot = pd.DataFrame({'date': df_close_stock['Date'],
                        'original_close':
df_close_stock['Close'],
                        'train_predicted_close':
train_predict_plot.reshape(1, -1)[0].tolist(),
                        'test_predicted_close':
test_predict_plot.reshape(1, -1)[0].tolist()})

fig = px.line(df_plot, x=df_plot['date'],
y=[df_plot['original_close'], df_plot['train_predicted_close'],

df_plot['test_predicted_close']],
              labels={'value': 'Stock price', 'date': 'Date'})
fig.update_layout(title_text='Comparison between original close
price vs predicted close price',
                  plot_bgcolor='white', font_size=15,
font_color='black', legend_title_text='Close Price')
fig.for_each_trace(lambda t: t.update(name=next(names)))

fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)
fig.show()

"""- # Прогнозирование на следующие 30 дней"""

x_input = test_data[len(test_data) - time_step:].reshape(1, -1)
temp_input = list(x_input)
temp_input = temp_input[0].tolist()

from numpy import array

lst_output = []
n_steps = time_step
i = 0
pred_days = 30
while (i < pred_days):

    if (len(temp_input) > time_step):
```

```
            x_input = np.array(temp_input[1:])
            #print("{} day input {}".format(i,x_input))
            x_input = x_input.reshape(1, -1)
            x_input = x_input.reshape((1, n_steps, 1))

            yhat = model.predict(x_input, verbose=0)
            #print("{} day output {}".format(i,yhat))
            temp_input.extend(yhat[0].tolist())
            temp_input = temp_input[1:]
            #print(temp_input)

            lst_output.extend(yhat.tolist())
            i = i + 1

    else:

            x_input = x_input.reshape((1, n_steps, 1))
            yhat = model.predict(x_input, verbose=0)
            temp_input.extend(yhat[0].tolist())

            lst_output.extend(yhat.tolist())
            i = i + 1

print("Output of predicted next days: ", len(lst_output))

"""- # Построение графика набора данных за последние 15 дней и
следующих прогнозируемых 30 дней"""

last_days = np.arange(1, time_step + 1)
day_pred = np.arange(time_step + 1, time_step + pred_days + 1)
print(last_days)
print(day_pred)

temp_mat = np.empty((len(last_days) + pred_days + 1, 1))
temp_mat[:] = np.nan
temp_mat = temp_mat.reshape(1, -1).tolist()[0]

last_original_days_value = temp_mat
next_predicted_days_value = temp_mat

last_original_days_value[0:time_step + 1] =
scaler.inverse_transform(df_close[len(df_close) -
time_step:]).reshape(1, -1).tolist()[0]
next_predicted_days_value[time_step + 1:] =
scaler.inverse_transform(np.array(lst_output).reshape(-1,
1)).reshape(1, -1).tolist()[0]

df_new_pred_plot = pd.DataFrame({
    'last_original_days_value': last_original_days_value,
    'next_predicted_days_value': next_predicted_days_value
})
```

```
names = cycle(['Last 15 days close price', 'Predicted next 30
days close price'])

fig = px.line(df_new_pred_plot, x=df_new_pred_plot.index,
y=[df_new_pred_plot['last_original_days_value'],

df_new_pred_plot['next_predicted_days_value']],
              labels={'value': 'Stock price', 'index':
'Timestamp'})
fig.update_layout(title_text='Compare last 15 days vs next 30
days',
                  plot_bgcolor='white', font_size=15,
font_color='black', legend_title_text='Close Price')

fig.for_each_trace(lambda t: t.update(name=next(names)))
fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)
fig.show()

"""- # Построение графика полной цены акций на момент закрытия с
прогнозированием на следующие 30 дней"""

df_lstm = df_close.tolist()
df_lstm.extend((np.array(lst_output).reshape(-1, 1)).tolist())
df_lstm = scaler.inverse_transform(df_lstm).reshape(1, -
1).tolist()[0]

names = cycle(['Close price'])

fig = px.line(df_lstm, labels={'value': 'Stock price', 'index':
'Timestamp'})
fig.update_layout(title_text='Построение графика полной цены
акций на момент закрытия с прогнозированием',
                  plot_bgcolor='white', font_size=15,
font_color='black', legend_title_text='Stock')
fig.for_each_trace(lambda t: t.update(name=next(names)))

fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)
fig.show()
```

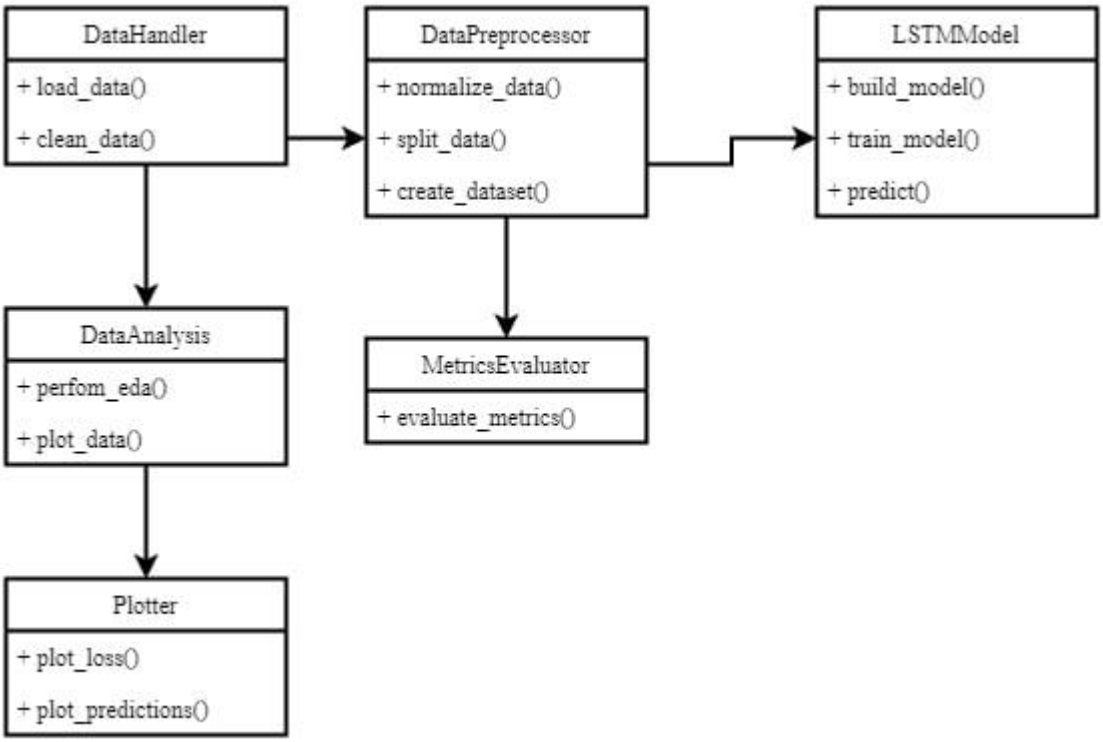
ПРИЛОЖЕНИЕ Б

Использование генетических алгоритмов в интеллектуальном анализе данных

UML –диаграммы

ПРИЛОЖЕНИЕ Б.1

Диаграмма классов



ПРИЛОЖЕНИЕ В

Использование генетических алгоритмов в интеллектуальном анализе данных

Презентация

ФГБОУ ВО «Пензенский государственный университет»
(Федеральное государственное бюджетное образовательное учреждение высшего образования)

Тема работы: «Использования генетических алгоритмов в интеллектуальном анализе данных»

Выполнил: Рудаков П.Д.
Научный руководитель: Валько А.Ф.

Слайд №1

План презентации

1. **Цели и задачи работы**
2. **Актуальность работы**
3. **Основные принципы генетических алгоритмов**
4. **Прогнозирование курса биткоина с помощью генетических алгоритмов**
5. **Анализ и предобработка исторических данных**
6. **Прогнозирование с помощью алгоритмов искусственного интеллекта**
7. **Преимущества и недостатки генетического алгоритма**
8. **Выводы**

Слайд №2

Цели и задачи работы

Цель работы:

Разработать метод прогнозирования курса биткоина с использованием генетических алгоритмов для повышения точности предсказаний

Задачи работы:

- Собрать и предобработать исторические данные о курсе биткоина.

- Разработать и настроить генетический алгоритм для оптимизации параметров модели прогнозирования.

- Построить и обучить модель нейронной сети LSTM, оптимизированную генетическим алгоритмом.

Слайд №3

Актуальность работы

Исследование прогнозирования курса биткоина с использованием генетических алгоритмов актуально из-за высокой волатильности криптовалютного рынка и недостатков традиционных методов прогнозирования.

Биткоин, как наиболее популярная криптовалюта, требует надежных прогнозов для минимизации рисков инвесторов.

Генетические алгоритмы предлагают инновационный подход, способный учитывать сложные зависимости и повышать точность прогнозов.

Слайд №4

Основные принципы генетических алгоритмов

Генетические алгоритмы основываются на принципах естественного отбора и генетики.

Эти алгоритмы имитируют процесс **эволюции**, создавая **популяцию возможных решений** и применяя к ним процессы **селекции, скрещивания и мутации**..

Слайд №5

Прогнозирование курса биткоина с помощью генетических алгоритмов

Применение генетических алгоритмов включает несколько этапов:

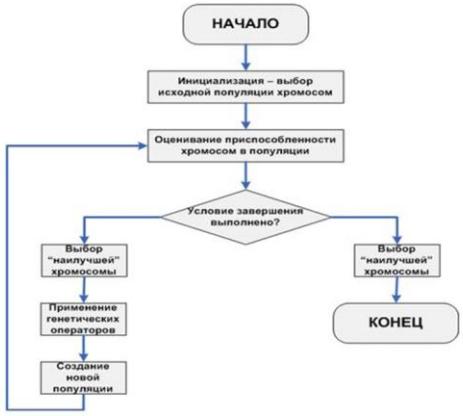
Инициализация популяции — создание начального набора решений.

Оценка приспособленности — определение качества каждого решения.

Селекция — отбор лучших решений для создания следующего поколения.

Скрещивание и мутация — создание новых решений путем обмена генетической информацией и внесения случайных изменений.

Преобразование популяции — обновление популяции новыми решениями.



Слайд №6

Анализ и предобработка исторических данных

Процесс предобработки данных включал:

Удаление выбросов — исключение аномальных значений.

Заполнение пропущенных значений — использование методов интерполяции и среднего значения.

Нормализация данных — приведение данных к единому масштабу.

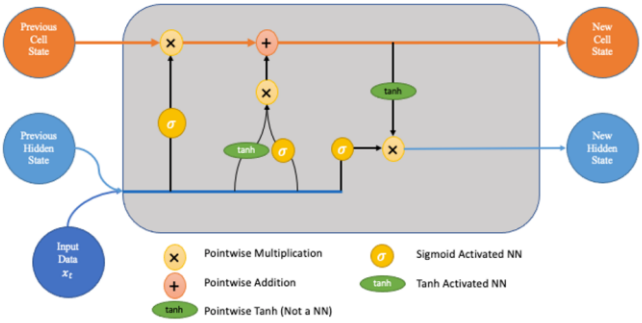
Анализ временных рядов — выявление трендов и сезонных колебаний.

Слайд №7

Прогнозирование с помощью алгоритмов искусственного интеллекта

Для прогнозирования курса биткоина мы использовали нейронные сети, в частности, модели LSTM (Long Short-Term Memory)

В процессе разработки модели мы использовали библиотеку DEAP для реализации генетического алгоритма, а также TensorFlow и Keras для построения и обучения нейронной сети



Слайд №8

Преимущества и недостатки генетического алгоритма

Преимущества:

- **Универсальность**: генетические алгоритмы могут работать с различными типами данных и учитывать множество факторов.
- **Гибкость**: возможность настройки и адаптации под конкретные задачи.
- **Эффективность**: способность обрабатывать большие объемы данных и находить оптимальные решения.

Недостатки:

- **Интерпретируемость**: сложность интерпретации результатов модели.
- **Скорость обучения**: необходимость большого количества итераций для достижения оптимального решения.
- **Зависимость от параметров**: эффективность алгоритма сильно зависит от выбранных параметров.

Слайд №9

Выводы

В заключение хочу отметить, что генетические алгоритмы и искусственный интеллект открывают новые возможности для анализа и прогнозирования в различных областях.

В будущем их применение будет только расширяться, улучшая точность и эффективность прогнозов.

Слайд №10

Спасибо за внимание!!!

Слайд №11