МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «БАЛТИЙСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ ИМЕНИ ИММАНУИЛА КАНТА»

Образовательно-научный кластер «Институт высоких технологий»

Высшая школа компьютерных наук и искусственного интеллекта

Выпускная квалификационная работа (магистерская диссертация)

Тема: Разработка рекомендательной системы для управления инвестиционным портфелем

Направление подготовки (специальность) 02.04.03 «Математическое обеспечение и администрирование информационных систем»
Профиль (специализация) «Банковские информационные технологии»

ОБУЧАЮЩЕГОСЯ _____ _____Н.А.Арифулина_____
(личная подпись) *(инициалы, фамилия)*

РУКОВОДИТЕЛЬ _____ к.т.н, доцент, С.Н.Ткаченко_____
(личная подпись) *(ученая степень, звание, инициалы, фамилия)*

РАБОТА ЗАЩИЩЕНА НА ОЦЕНКУ _____

Допустить к защите

Директор высшей школы _____ PhD, доцент, Верещагин М.Д.
(личная подпись) *(ученая степень, звание, инициалы, фамилия)*

“___” _____ 20___ г.

Калининград – 2024 г.

Реферат

Выпускная квалификационная работа содержит 66 страниц, 34 рисунка, 3 таблицы, 15 используемых источников.

Цель работы – разработка рекомендательной системы для управления инвестиционным портфелем.

Задача была решена за счет использования знаний и навыков, полученных в результате обучения в магистратуре. В работе сделан акцент на обучении алгоритма обучения с подкреплением, используя нейронные сети. Задача заключалась в создании отдельной рекуррентной нейронной сети для каждой акции и использовании их предсказаний для основы работы алгоритма распределения активов. Кроме того, целью было интегрировать все вышеперечисленные модели в приложение, добавив в него различные визуальные элементы.

Первая глава работы посвящена исследованию современных подходов прогнозирования цен акций с помощью нейронных сетей.

Вторая глава содержит первичный анализ и обработка исходных данных, процесс подбора и обучения рекуррентных нейронных сетей и обучения алгоритмов обучения с подкреплением.

В третьей главе описываются результаты разработанной системы и процесс разработки приложения.

В заключении сделан вывод по работе в соответствии с поставленными задачами.

Оглавление

Введение

Задача формирования инвестиционного портфеля требует от аналитика больших усилий. Это связано с большим количеством инструментов, которые существуют на рынке, и случайным характером изменения стоимости активов.

В этой связи различные приложения предоставляют множество сервисов для биржевых брокеров. Наиболее востребованные из них связаны с визуализацией данных и прогнозированием стоимости различных активов, что и позволяет брокеру формировать инвестиционный портфель с наибольшей эффективностью.

Поскольку все большее количество пользователей получает доступ к биржевым инструментам, то возникает потребность в разработке подобных инструментов для широкого круга лиц, поскольку все большее количество людей имеют потребность в формировании личного инвестиционного портфеля.

Основу подобного приложения должна составлять рекомендательная система, обладающая следующими свойствами:

- Динамический анализ различных биржевых бумаг;
- Агент для формирования инвестиционного портфеля;
- Удобный и понятный интерфейс.

В связи с чем, целью выпускной квалификационной работы является разработка рекомендательной системы для управления инвестиционным портфелем.

Для достижения этой цели были поставлены следующие задачи:

1. Обзор современных подходов к управлению инвестиционным портфелем от классических, до основанных на искусственном интеллекте.
2. Непосредственная разработка рекомендательной системы с использованием анализа временных рядов и самообучающегося агента.

3. Тестирование разработанной рекомендательной системы на модельном портфеле и сравнение полученных результатов со среднерыночными.

Объектом исследования являются финансовые временные ряды и методы их анализа для принятия инвестиционных решений.

Предметом исследования выступают алгоритмы машинного обучения и нейронные сети, применяемые для управления инвестиционными портфелями.

Глава 1. Основы управления инвестиционным портфелем и прогнозирования цен акций

1.1 Основы управления инвестиционным портфелем

Современный этап развития общества характеризуется постоянными изменениями окружающей среды и свободной конкуренцией на рынке капиталов за привлечение инвестиций. В условиях такой конкуренции каждая компания стремится привлечь как можно больше инвестиций. С другой стороны, современному инвестору предстоит оценка привлекательности компаний, принятие правильных инвестиционных решений и оценка эффективности своих инвестиций.

Инвестиционный портфель представляет собой совокупность финансовых и реальных активов, подобранных инвестором в различных пропорциях с целью получения максимальной прибыли или диверсификации рисков. Состав и количество активов в портфеле зависят от опыта и интересов инвестора.

Актуальность проблемы проявляется в том, что в условиях экономического кризиса эффективное управление инвестиционной деятельностью как в масштабах государства, так и в рамках отдельных хозяйствующих субъектов становится одним из важнейших вопросов. В практике портфельного инвестирования выделяются три основных вида портфелей ценных бумаг, классифицируемых в зависимости от отношения инвестора к риску [1].

Консервативный портфель состоит из низкорисковых финансовых инструментов, таких как государственные и муниципальные облигации, депозиты, драгоценные металлы и недвижимость. Хотя консервативный портфель не приносит большой доходности, он минимизирует риски. Средняя доходность составляет 4–12% в год. Такой портфель выбирают новички или люди, которые инвестируют на длительный срок и хотят

сохранить денежные средства от инфляции. Он надежен и не требует активного управления. Обычно такие портфели состоят из: 50-70% различных облигаций, 10-20% банковские вклады, до 10-20% драгоценные металлы, до 10% недвижимость.

Сбалансированный портфель включает низкорисковые и более рисковые активы, такие как акции и фонды крупных компаний, а также валюта. Доходность сбалансированного портфеля выше, но и риск тоже выше. Такой портфель может иметь следующие доли классов активов: 20-40% акции первого эшелона или голубых фишек, 10-30% облигации, 10-30% индексные фонды, до 10% различные валюты.

Агрессивный портфель включает высокодоходные и рисковые активы, такие как акции, опционы и бумаги компаний, недавно размещенных на бирже. Риск потери инвестиций в агрессивном портфеле выше. Инвесторы, которые хотят максимизировать свою прибыль и готовы следить за рынком ежедневно, выбирают агрессивный подход. У таких портфелей наибольшую долю имеют класс активов акции (50-70%), фьючерсы и опционы (20-30%), другие различные высокорисковые активы (10-20%).

После формирования портфеля ценных бумаг важно сохранить его основные инвестиционные качества, соответствующие интересам держателя. Для этого необходимо разработать и придерживаться своей стратегии управления портфелем ценных бумаг.

Стратегия управления портфелем ценных бумаг представляет собой совокупность методов и способов воздействия на повышение эффективности портфеля, включая использование различных математических моделей и технических возможностей. Выделяют два основных типа стратегий: активную и пассивную.

Рис.1. Управление портфелем ценных бумаг

Активная стратегия управления портфелем характеризуется постоянным изменением состава и структуры портфеля с целью получения большей доходности за счет волатильности рынка. Инвесторы, которые придерживаются активной стратегии, обычно не верят в постоянную эффективность рынка и считают, что способны обеспечить более эффективную структуру портфеля и получить более высокий доход благодаря своей информированности. Активный стиль управления является весьма трудоемким и требует значительных трудовых и финансовых затрат, т.к. он связан с активной аналитической, информационной и торговой деятельностью на финансовом рынке. Как правило, активным управлением портфелем занимаются крупные профессиональные участники финансового рынка, например банки, инвестиционные фонды и др., обладающие

большими финансовыми возможностями и штатом высококвалифицированных менеджеров (управляющих портфелями) и отделами аналитиков.

Пассивная стратегия управления портфелем предполагает формирование изначально хорошо диверсифицированного портфеля с определенными показателями доходности и риска на долгосрочную перспективу и его дальнейшее держание без значительных изменений. Пассивные портфели основываются на предположении об эффективности рынка и одинаковых ожиданиях инвесторов относительно доходности и риска. Самым распространенным методом в пассивной стратегии управления портфелем ценных бумаг является метод индексного фонда, т.е. попытка инвестора «купить рынок». Пассивные портфели характеризуются низким оборотом, минимальным уровнем накладных расходов и низким уровнем специфического риска.

Таким образом, управление портфелем ценных бумаг зависит от выбранной инвестором стратегии. Для достижения целей инвестирования, рационального использования ресурсов и адекватного управления рисками важно выбрать эффективную модель управления, включающую определенные этапы.

1.2 Прогнозирование цен акций

Существует несколько концепций о возможности прогнозирования динамики фондовых рынков. Одной из них является гипотеза эффективного рынка, которая утверждает, что цены акций уже содержат всю доступную информацию, и поэтому прогнозирование становится бессмысленным. Эта гипотеза находит продолжение в теории случайных блужданий.

В рамках теории случайных блужданий информация делится на предсказуемую, известную, и новую, неожиданную. Если предсказуемая информация уже учтена в ценах акций, то новая неожиданная информация

еще не отражена в цене. Одной из характеристик неожиданной информации является ее случайный характер, что приводит к случайным изменениям цен. Гипотеза эффективного рынка объясняет изменения цен при поступлении новой информации, в то время как теория случайных блужданий дополняет это предположение о случайности изменений цен.

В сущности, теория случайных блужданий советует применять стратегию "покупай и держи". Эта концепция стала особенно популярной в 70-е годы, когда на американском фондовом рынке не было четких тенденций, а рынок находился в узком диапазоне. Согласно гипотезе эффективного рынка и теории случайных блужданий, прогнозирование цен акций является невозможным.

Однако, несмотря на это, многие участники рынка всё же пытаются применять различные методы для прогнозирования, предполагая, что временные ряды скрывают скрытые закономерности. В 30-е годы основатель технического анализа Р. Эллиот пытался выявить такие скрытые закономерности во временных рядах, что нашло отражение в опубликованных статьях [2]. В 80-е годы эта точка зрения получила дополнительную поддержку в теории динамического хаоса.

В настоящее время профессиональные участники рынка используют различные методы прогнозирования финансовых временных рядов, основные из них:

1) Экспертные методы прогнозирования.

Самый распространенный метод из группы экспертных методов — метод Дельфи. Его суть заключается в агрегации мнений экспертов для получения общей оценки ситуации на рынке. При использовании этого метода необходимо сформировать группу экспертов, компетентных в данной области, провести опрос и сделать выводы о текущем состоянии рынка.

2) Методы логического моделирования.

Основаны на поиске и выявлении закономерностей рынка в долгосрочной перспективе. Сюда включаются метод сценариев, метод прогнозов по образу и метод аналогий.

3) Экономико-математические методы.

Эти методы основаны на создании моделей объекта исследования. Очень важным для финансовой науки является использование оптимизационных моделей, которые представляют собой систему уравнений с ограничениями и функционалом оптимальности для нахождения наилучшего решения.

4) Статистические методы.

Статистические методы прогнозирования применительно, для финансовых временных рядов основаны на построении различных индексов (диффузный, смешанный), расчет значений дисперсии, мат ожидания, вариации, ковариации, интерполяции, экстраполяции.

5) Технический анализ.

Этот метод предполагает прогнозирование изменений цен на основе анализа исторических данных о ценах. Он основан на выявлении трендов во временных рядах цен.

6) Фундаментальный анализ.

Этот метод используется для оценки рыночной стоимости компании на основе анализа её финансовых и производственных показателей. Этому анализу подвергаются финансовые показатели компании: выручка, EBITDA, чистая прибыль, долговые обязательства, денежный поток, величина выплачиваемых дивидендов и производственные показатели компании.

Использование нейронных сетей для прогнозирования финансовых временных рядов также можно отнести к техническому анализу. Эти сети пытаются выявить закономерности в развитии временных рядов на основе их исторических данных. Однако финансовые временные ряды зачастую содержат значительный уровень шума, поэтому перед использованием

нейронных сетей необходимо провести тщательную предобработку данных и кодирование переменных.

1.3 Подход к прогнозированию временных рядов с помощью нейронных сетей

Исследования, посвященные применению нейронных сетей для прогнозирования временных рядов, имеют длительную историю развития. Начиная с пионерских работ в области нейронных сетей в конце 20-го века, исследователи активно исследовали и адаптировали различные архитектуры нейронных сетей для прогнозирования различных типов временных рядов.

Важным моментом таких исследований является сравнительный анализ различных архитектур нейронных сетей для временных рядов. Различные архитектуры, такие как простые рекуррентные нейронные сети (RNN), Long short-term memory (LSTM) и Gated Recurrent Units (GRU), имеют свои преимущества и недостатки в контексте прогнозирования временных рядов. Исследователи стремятся выявить оптимальные архитектуры и параметры нейронных сетей для достижения наилучшей производительности в прогнозировании.

Однако, несмотря на значительные успехи, существуют проблемы и ограничения в прогнозировании финансовых временных рядов с использованием нейронных сетей, более подробно часть из них мы рассмотрим дальше в работе. В частности, финансовые временные ряды часто характеризуются высокой степенью шума и не стационарностью, что может затруднить точное прогнозирование. Кроме того, выбор оптимальной архитектуры и параметров нейронных сетей, а также эффективная обработка данных, остаются вызовами для исследователей в этой области.

В статье [3] авторы приводят обзор большого количества многочисленные исследования, посвященные применению нейронных сетей в прогнозировании. Они отмечают, что нейронные сети обладают

уникальной способностью адаптироваться к различным задачам и могут аппроксимировать нелинейные зависимости данных, что подчеркивает потенциал использования нейронных сетей в задачах прогнозирования временных рядов.

Рассмотрим более подробно про рекуррентные нейронные сети.

Рекуррентные нейронные сети (RNN, РНС) представляют собой класс нейронных сетей, полезных для моделирования последовательных данных. Произошедшие от прямых нейронных сетей, РНС проявляют поведение, схожее с тем, как функционирует человеческий мозг. Проще говоря, рекуррентные нейронные сети обеспечивают предсказательные результаты в последовательных данных, чего не могут другие алгоритмы.

Чтобы понять принцип работы РНС, нужно иметь начальные знания о нейронных сетях прямой связи и последовательных данных.

Последовательные данные — это упорядоченные данные, в которых связанные вещи следуют друг за другом. Примерами таких данных являются финансовые данные или последовательность ДНК. Самый популярный тип последовательных данных временные ряды данных, представляющие собой просто серию точек данных, упорядоченных во времени.

РНС и нейронные сети прямой связи получили свои названия от способа передачи информации. В прямой нейронной сети информация движется только в одном направлении — от входного слоя через скрытые слои к выходному слою. Информация движется прямо через сеть.

Прямые нейронные сети не имеют памяти о входных данных и плохо предсказывают, что будет дальше. Поскольку прямая сеть учитывает только текущие входные данные, у нее нет понятия порядка во времени. Она просто не может помнить что-либо о том, что произошло в прошлом, кроме своего обучения.

В RNN информация проходит через цикл. Когда она принимает решение, она учитывает текущий ввод, а также то, что она узнала из предыдущих входных данных.

На рисунке 2 показана разница в потоке информации между рекуррентной и прямой нейронной сетью.

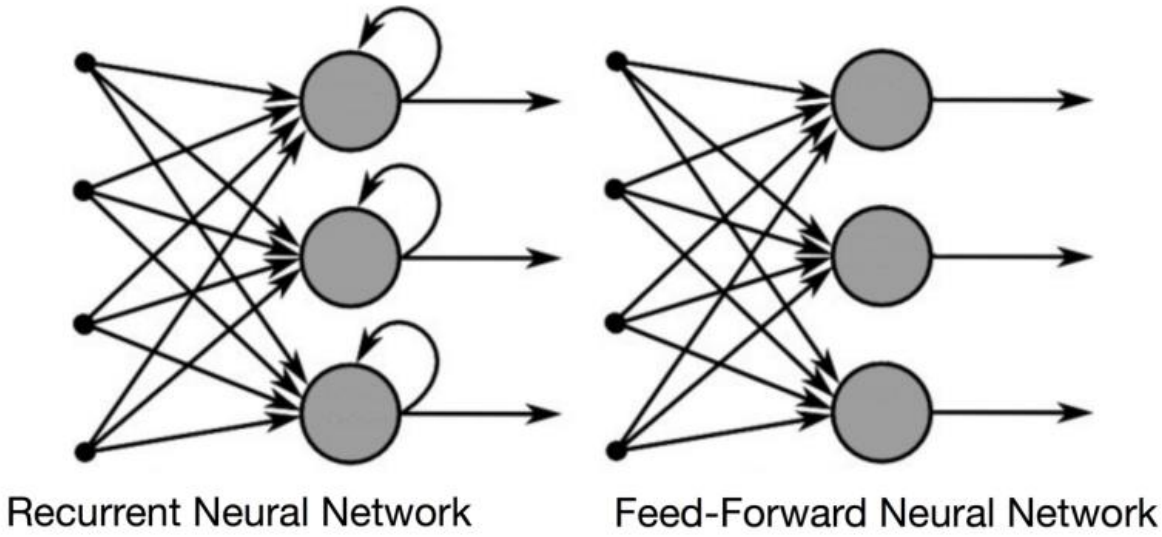

Рис.2. Отличия рекуррентной нейронной сети от прямой

Для лучшего понимания можно рассмотреть следующий пример. Допустим, что есть обычная прямая нейронная сеть, и дав ей слово «нейрон» в качестве входных данных, и она обрабатывает слово по символам. К тому времени, как она достигнет символа «р», она уже забыла о «н», «е» и «у», что делает практически невозможным для этого типа нейронной сети предсказать, какой символ будет следующим. Рекуррентная нейронная сеть способна запомнить эти символы благодаря своей внутренней памяти. Она создает выходные данные, копирует их и циклически возвращает в сеть.

Таким образом, у RNN два входа: настоящее и недавнее прошлое. Это важно, потому что последовательность данных содержит важную информацию о том, что будет дальше, что позволяет РНС выполнять задачи, которые не могут другие алгоритмы.

Прямая нейронная сеть, как и все другие алгоритмы глубокого обучения, присваивает своим входным данным весовую матрицу и затем создает выходной результат. В то время РНС применяют веса к текущим, а

также и к предыдущим входным данным. Более того, рекуррентная нейронная сеть также будет корректировать веса как для градиентного спуска, так и для обратного распространения ошибки во времени.

Существуют несколько типов RNN:

- Один ко одному
- Один ко многим
- Многие к одному
- Многие ко многим

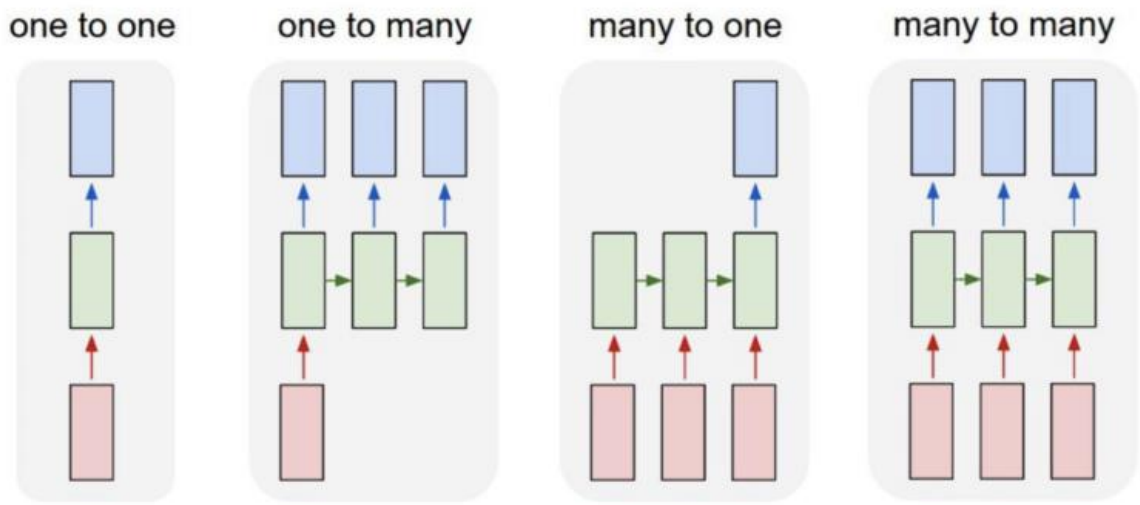

Рис.3. Типы RNN

Рекуррентные нейронные сети нашли широкое применение в таких областях, как обработка естественного языка (включая задачи машинного перевода, распознавания речи и генерации текста), прогнозирования временных рядов (прогнозирование финансовых показателей, температуры и других параметров) [3]. Кроме того, они часто используются в анализе данных, медицинской диагностике, обработке аудио и видео, искусственном интеллекте и других областях, где присутствует последовательная структура данных.

К преимуществам РНС можно отнести способность учитывать контекст и зависимости между элементами последовательности, а также возможность

обработки данных произвольной длины. Это позволяет им эффективно работать с данными различных временных масштабов и автоматически извлекать признаки из последовательных данных.

Но у рекуррентных нейронных сетей есть и свои недостатки. Одной из основных проблем является слабая способность моделировать долгосрочные зависимости из-за проблемы исчезающего и взрывного градиента, который мы рассмотрим ниже.

При обучении РНС возникают трудности, связанные с рекуррентными связями между слоями. В частности, проблемы могут возникнуть при использовании алгоритма обратного распространения ошибки, особенно при разворачивании сети во времени, которое называется обратным распространением ошибки сквозь время (Backpropagation through time) [4]. Они также склонны к забыванию далеко расположенных контекстов, что может привести к потере информации о предыдущих состояниях.

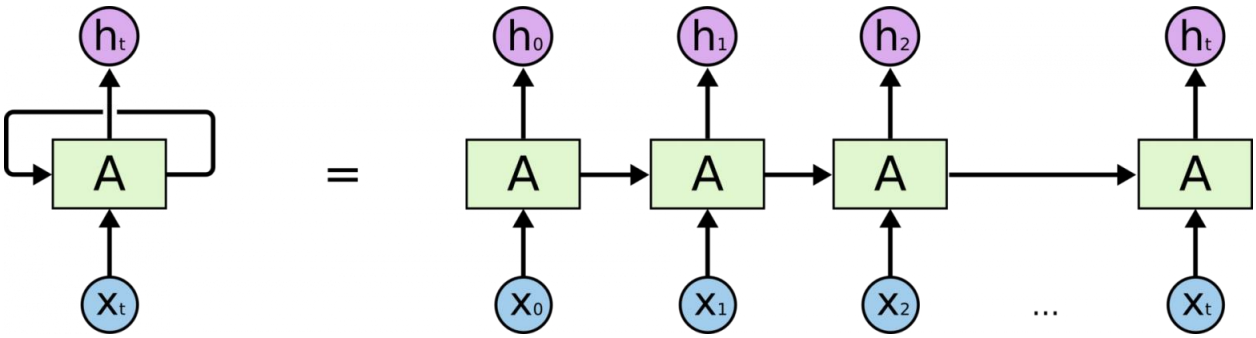

Рис.4. Рекуррентная нейронная сеть в развертке

Развернутая таким образом сеть может стать очень глубокой, что создает проблемы с вычислением градиента. Это может привести к проблеме исчезающего градиента, когда веса становятся очень маленькими, или к проблеме взрывного градиента, когда веса становятся очень большими. Для решения этих проблем часто используются специализированные типы нейронных блоков, такие как Long short-term memory (LSTM) и Gated Recurrent Units (GRU), способные эффективно управлять потоком информации и минимизировать проблемы с градиентом.

В контексте прогнозирования временных рядов, блок LSTM привлекает особое внимание и широко используется исследователями. Так, в статье [4] проведен сравнительный анализ между LSTM, GRU и ERNN, по результатам которого блок LSTM показал наилучшие показатели метрик. Его способность к эффективной работе с последовательными данными и управлению долгосрочными зависимостями делает его предпочтительным выбором.

Блоки LSTM были предложены в 1997 году Хохрайтером и Шмидхубером. Они широко используются в задачах обработки текстовой информации, таких как машинный перевод и генерация текста. LSTM — это рекуррентный нейрон, разработанный для обучения долгосрочным зависимостям в данных. Его уникальность заключается в наличии блока памяти, способного сохранять активацию бесконечно. Этот блок состоит из линейных нейронов, которые сохраняют текущее состояние памяти, и трех вентилей, контролирующих поток информации.

Вентили LSTM реализованы в виде логистической функции, вычисляющей значения в диапазоне [0; 1]. Они регулируют поток информации внутри и вне блока памяти. Например, "входной вентиль" управляет тем, как новая информация входит в память, "вентиль забывания" решает, какие значения сохранить, а "выходной вентиль" контролирует использование значений из памяти для вычисления выхода блока.

Визуализация нейрона LSTM представлена на рисунке 5.

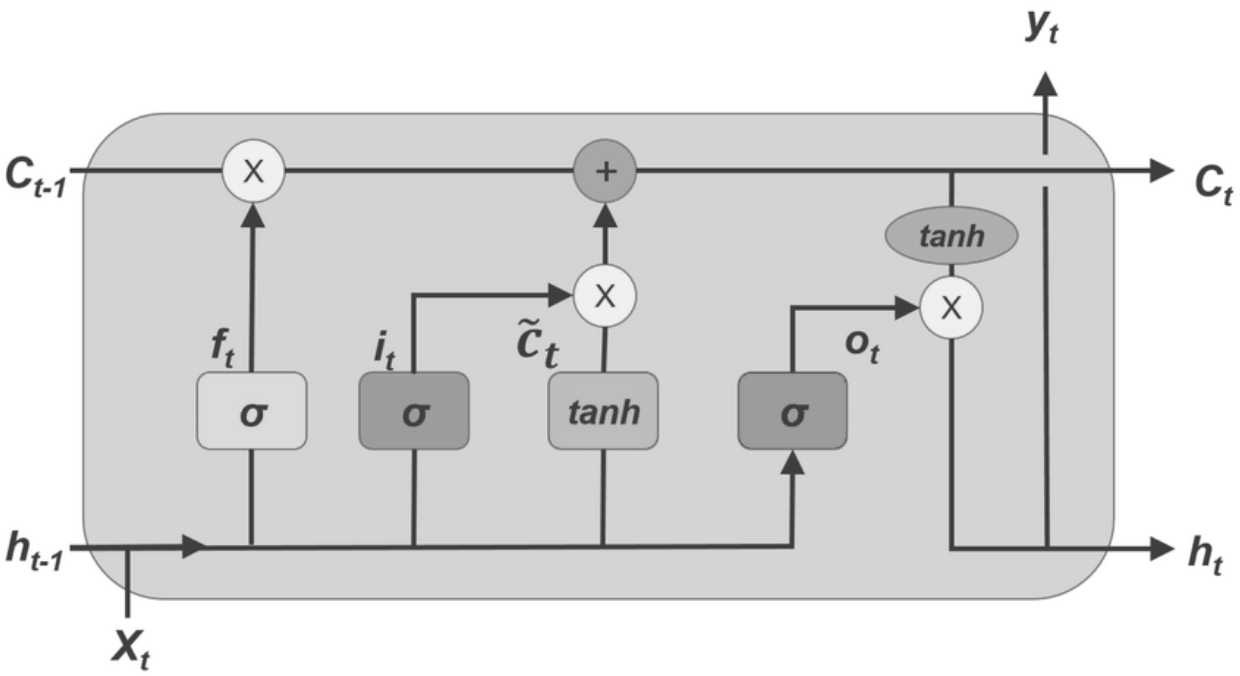

Рис.5. Визуализация нейрона LSTM

Каждый LSTM нейрон задается следующими формулами:

$$f_t = \sigma_g\left(w_f x_t + U_f * h_{t-1} + b_f\right)$$

$$i_t = \sigma_g(w_i x_t + U_i * h_{t-1} + b_i)$$

$$o_t = \sigma_g(w_o x_t + U_o * h_{t-1} + b_o)$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \sigma_c(w_c x_t + U_c * h_{t-1} + b_c)$$

$$h_t = o_t \circ \sigma_h(c_t)$$

$$c_0 = 0,\ h_0 = 0,$$

где

- x_t -входной вектор;
- h_t -выходной вектор;
- c_t- вектор состояний;
- W, U -матрицы весов;
- b - вектор свободного члена;
- f_t - вектор вентиля забывания, вес запоминания старой информации;
- i_t - вектор входного вентиля, вес получения новой информации;
- o_t- вектор выходного вентиля.

Идея заключается в том, что LSTM может забывать старые значения, когда появляются новые, что помогает в обработке долгосрочных зависимостей [5]. В задачах прогнозирования временных рядов это свойство особенно ценно, поскольку последние значения обычно имеют большее влияние на прогноз модели.

Хотя у LSTM есть проблемы с памятью из-за подавления старых значений новыми, в контексте прогнозирования временных рядов это свойство может быть полезным для выявления влияния последних данных на прогноз.

В работе использовался не простой блок LSTM, а его вариация. Bidirectional LSTM (BiLSTM) представляет собой модификацию классического LSTM блока, которая позволяет учитывать как предшествующие, так и последующие значения во временном ряду или последовательности данных. Основное отличие BiLSTM от обычного LSTM заключается в том, что BiLSTM обрабатывает входные данные как в прямом, так и в обратном направлении. Это позволяет модели учитывать контекст как перед текущим моментом времени, так и после него.

Использование BiLSTM особенно полезно в задачах, где контекст представляет собой важный аспект данных. В задачах прогнозирования временных рядов BiLSTM может улучшить способность модели улавливать сложные зависимости в данных, учитывая как прошлые, так и будущие значения.

Схема работы BiLSTM представлен на рисунке 6.

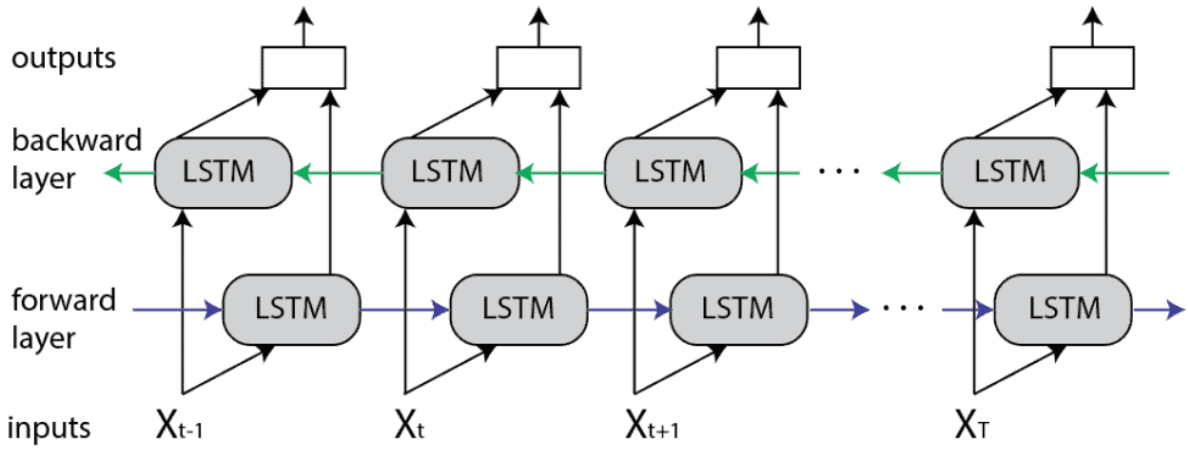

Рис.6. Развернутый BiLSTM слой

Существует несколько практических подходов к прогнозированию с использованием рекуррентных нейронных сетей:

- прогнозирование одномерных временных рядов;
- прогнозирование многомерных временных рядов.

В первом случае в качестве прогнозных факторов используются только предыдущие значения временного ряда. Во втором случае к ним могут добавляться лаги дополнительных рядов, которые выступают в качестве прогнозных факторов, от которых, по предположению исследователя, зависит целевой показатель.

В работе [4] отмечается, что глобальные модели рекуррентных нейронных сетей часто сталкиваются с выбросами в отдельных временных рядах. Предположительно, это происходит из-за того, что средние веса нейронной сети, определенные путем подгонки глобальных моделей, могут быть неоптимальными для индивидуальных требований различных временных рядов. Таким образом, возникает потребность в разработке новых моделей, которые бы включали как глобальные, так и локальные параметры для каждого временного ряда, представленные в виде иерархических моделей. Такие модели могут быть потенциально комбинированы с ансамблями, где каждая модель обучается по-разному на имеющемся наборе данных.

Работы [6] и [7] являются шагами в этом направлении, но эта тема остается значительно открытым исследовательским вопросом. K.Bandara также отмечает [6], что плохая эффективность моделей машинного обучения в прогнозировании одномерных временных рядов на соревнованиях, вероятно, связана с короткой длиной временных рядов, что делает их неподходящими для сложных моделей нейронных сетей. В таких случаях более простые статистические модели, устойчивые к шуму, как правило, показывают хорошие результаты.

K.Bandara предлагает решить проблему накопления ошибок при переходе к глобальным моделям, построив отдельную нейронную сеть для каждого кластера похожих временных рядов. В их основе лежит рекуррентная нейронная сеть со слоем LSTM.

Авторы также уделяют внимание предобработке временных рядов, рекомендуя удалить сезонность, логарифмировать ряды, использовать преобразование Бокса-Кокса и нормализовать временные ряды. В заключении статьи [6] авторы делают вывод о том, что глобальные модели превосходят локальные модели по качеству, и стратегия построения отдельной модели для каждого кластера временных рядов дает лучшие результаты.

Комплексной обзорной статьей можно назвать работу [8], в которой исследователи рассматривают различные виды нейронных сетей применимо к финансам. В ней рассматриваются различные модели глубокого обучения, такие как Deep Multilayer Perceptron (DMLP), RNN, LSTM, Convolutional Neural Network (CNN), Deep Belief Network (DBN), Restricted Boltzmann Machines (RBMs), Deep Reinforcement Learning (DRL), Autoencoder (AE), и их применение в прогнозировании финансовых временных рядов. Авторы исследуют несколько подобластей, включая прогнозирование цен на акции, биржевых индексов, облигаций, сырьевых товаров и криптовалют.

Также отмечается, что публикации классифицируются на две группы в зависимости от сущности выхода модели: прогнозирование цены или

предсказание тренда временного ряда. Вторая группа иногда рассматривается как дополнительная задача для первой, поскольку для предсказания направления тренда необходимо предсказать сами значения временного ряда. Некоторые исследователи также рассматривают эту задачу как самостоятельную и решают ее как задачу классификации, обходя тем самым задачу прогнозирования.

Авторы делают акцент, что LSTM и его вариации, наряду с некоторыми гибридными моделями, преобладают в сфере прогнозирования финансовых временных рядов, благодаря их способности хорошо выделять временные характеристики рядов. Некоторые исследователи также рекомендуют использовать нормализацию для временных рядов, удалять сезонные составляющие или преобразовывать ряды для обеспечения стационарности.

Также в своей статье авторы приводят статистику публикаций, которая показывает, что более половины работ по прогнозированию временных рядов относится к моделям RNN. Анализ публикаций подтверждает доминирование рекуррентных нейронных сетей, в основном состоящих из блоков LSTM. Независимо от типа задачи, будь то прогнозирование цен или направления тренда, RNN, GRU и LSTM рассматриваются как основные модели, часто используемые для сравнительного анализа с другими методами.

В статье делается вывод, что большинство реализаций RNN для прогнозирования финансовых временных рядов осуществляется на Python с использованием библиотек Keras и Tensorflow. Это указывает на растущую потребность в разработке прогностических систем с использованием RNN на этом языке.

В данной работе мы будем ограничиваться областью прогнозирования цен на акции. Авторы статьи [8] называют эту область одной из самых популярных среди финансовых временных рядов. Также описывается использование LSTM моделей для прогнозирования котировок с использованием различных макроэкономических показателей,

интеллектуального анализа новостей, показателей технического анализа и других данных. Это отражает интерес к изучению моделей прогнозирования многомерных временных рядов, где статистические методы оказываются менее эффективными.

Авторы также отмечают, что в большинстве исследований модели нейронных сетей оказываются точнее моделей машинного обучения. Однако иногда их показатели сопоставимы. Это может быть связано с растущей популярностью нейронных сетей, доступностью данных и возможностью аппроксимации нелинейных функций.

Также есть ряд работ российских авторов по данному направлению, которые рассмотрены ниже.

В статье [9] проведен сравнительный анализ моделей LSTM и ARIMA в прогнозировании курса биткоина к доллару. В работе использовались дневные данные с января 2015 года по декабрь 2018 года. Авторы сделали прогноз на короткий прогнозный горизонт (1 период вперед), без дополнительной предобработки временного ряда. Полученное качество прогноза оказалось достаточным для рекомендации использования этих моделей в задачах прогнозирования котировок криптовалюты на короткие временные интервалы.

Следующие исследователи рассмотрели [10] 36 различных конфигураций архитектур LSTM-сетей для построения прогнозов длительностью до 70 шагов по данным, размер которых составляет 300–500 элементов. В качестве данных использовались математическое ожидание, дисперсия, коэффициенты асимметрии и эксцесса смесей плазмы в разные моменты времени. Временные ряды были нормализованы и разделены методом скользящего окна.

Авторы сравнили архитектуры с одним, двумя и тремя скрытыми слоями LSTM и пришли к выводу, что архитектура с двумя скрытыми слоями дает наибольший прирост точности. В среднем, архитектура с двумя слоями дает на 18% меньшую ошибку RMSE и на 20% меньшую ошибку MAE. Они

также установили, что эффект от использования архитектуры с двумя скрытыми слоями менее заметен, если отношение между прогнозным горизонтом и входным окном меньше 0.1.

Далее в статье [11] авторы решают кейс прогнозирования количества заявок в крупную федеральную информационную систему. Они пришли к выводу, что наименьшая ошибка наблюдается у нейронных сетей, которые тренировались с оптимизаторами Adam и Nadam и имели 20 LSTM блоков, причем рассматривалась архитектура только с одним скрытым слоем.

Наконец, в статье [12] сравнивается качество прогнозов для моделей ARIMA и LSTM на основе различных котировок акций российских компаний за период с июня 2014 по ноябрь 2019 года с разбивкой по неделям. Результаты показывают превосходство модели LSTM, где среднеквадратическая ошибка RMSE на 65% меньше, чем у модели ARIMA.

В целом, глубокое обучение — это быстроразвивающаяся область исследований, где появляется множество новых архитектур и методов. Однако для разработчиков прогнозных систем остается вопрос, в каких ситуациях эти методы наиболее полезны и насколько сложно их адаптировать к конкретным сценариям применения.

Глава 2. Разработка рекомендательной системы на основе прогнозов цен акций

2.1 Используемый стек технологий

Рекомендательная система для управления инвестиционным портфелем строится на основе нескольких важных компонентов, каждый из которых играет ключевую роль в обеспечении эффективного функционирования системы:

1. Интерфейс пользователя: Этот компонент обеспечивает пользователю удобный и интуитивно понятный интерфейс для взаимодействия с системой. Веб-приложение/сервис на фреймворке Streamlit позволяет пользователям легко задавать параметры своего портфеля, просматривать рекомендации по его составлению и аналитическую информацию о своих инвестициях.

2. Прогнозирование цен акций: Этот компонент используется для анализа временных рядов и прогнозирования цен акций. RNN обучаются на исторических данных о ценах акций и способны предсказывать их будущие изменения, что является основой для принятия решений о составе портфеля.

3. В компоненте «Распределение активов в портфеле» используется реализация алгоритма с подкреплением, с помощью которого собирается оптимальный портфель из предсказанных акций. Задача этого алгоритма заключается в максимизации ожидаемой прибыль и минимизации рисков.

4. С помощью базы данных и хранилища данных реализовано хранение и управление такими данными как исторические цены акций, различные параметры портфелей и другой информации. Благодаря этому обеспечивается быстрый доступ к необходимым данным и их сохранность и целостность в течение всей сессии использования системы.

При выборе технологий и инструментов для разработки рекомендательной системы было уделено особое внимание следующим аспектам:

1. Python: в качестве основного языка программирования был выбран Python, благодаря его простоте, гибкости и богатому экосистему библиотек для машинного обучения и анализа данных.

2. Streamlit: Фреймворк Streamlit был выбран для создания веб-приложения/сервиса, так как он обладает простым синтаксисом и позволяет быстро создавать интерактивные приложения на языке Python.

3. Keras и TensorFlow: Библиотеки Keras и TensorFlow были выбраны для реализации рекуррентных нейронных сетей и алгоритма DQN. Они обеспечивают высокую производительность и гибкость при создании и обучении нейронных сетей.

4. Stable Baselines3: Популярная библиотека для обучения агентов с использованием методов обучения с подкреплением (RL). Библиотека включает в себя реализацию множества популярных алгоритмов обучения с подкреплением, таких как PPO (Proximal Policy Optimization), DQN (Deep Q-Networks), A2C (Advantage Actor-Critic), TRPO (Trust Region Policy Optimization) и других, а совместимость с OpenAI Gym позволяет использовать широкий спектр стандартных сред для тестирования и обучения агентов.

5. SQLAlchemy и SQLite: для работы с базой данных были выбраны библиотека SQLAlchemy для взаимодействия с базой данных на уровне объектов и база данных SQLite для хранения и управления данными.

Полная архитектура проекта, его компоненты и связи между ними представлены на рисунке 7.

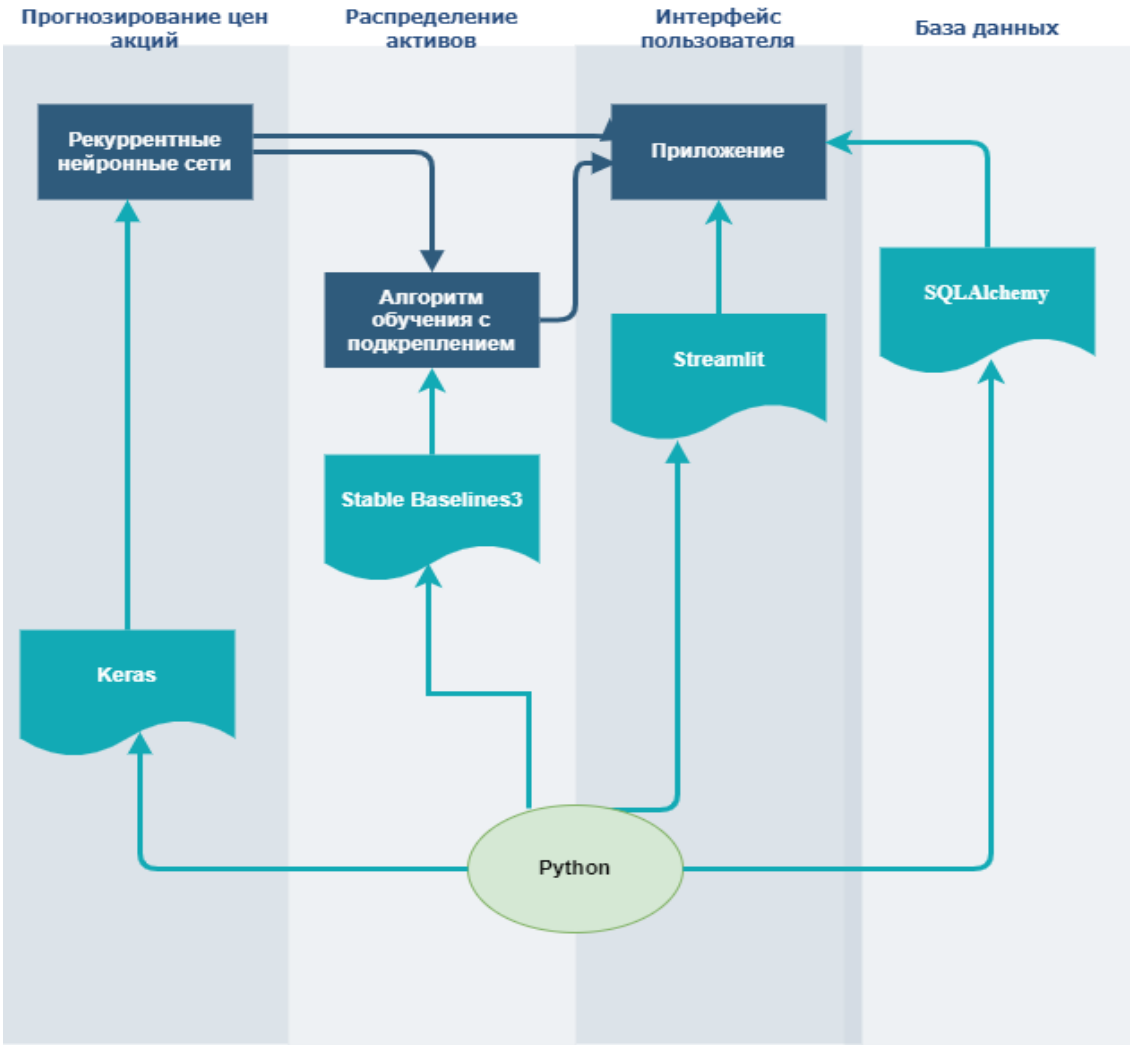

Рис.7. Полная архитектура проекта

Выбор данных технологий и инструментов обеспечивает эффективную разработку, высокую производительность и качество реализации рекомендательной системы для управления инвестиционным портфелем.

2.2 Сбор, предобработка и анализ данных

Для обучения и тестирования нейронных сетей, был собран набор данных с акциями с помощью специализированной библиотеки московской биржи на Python apimoex за период с 01-01-2015 по 01-04-2024. Учитывая огромное разнообразие акций на бирже, необходимо было определиться с набором акций, цены которых будут предсказываться. Для этого решено было использовать акции компаний малой и средней капитализации из

индекса MCXSM. Акции этой категории обладают большой волатильностью, за счет которой у нас будет возможность получить доходность выше средней по всему рынку. Компании представлены в таблице 1.

Таблица 1. Выбранные компании

Тикер	Компания	Отрасль
AFKS	АФК Система	Финансовый сектор
AFLT	Аэрофлот	Транспорт
AGRO	РусАгро	Потребительские товары и услуги
APTK	Аптечная сеть 36,6	Потребительские товары и услуги
AQUA	ИНАРКТИКА/Русская аквакультура	Потребительские товары и услуги
BELU	Белуга	Потребительские товары и услуги
BSPB	Банк Санкт-Петербург	Финансовый сектор
CBOM	Московский кредитный банк	Финансовый сектор
CIAN	Циан	Информационные технологии
ELFV	ЭЛ5-Энерго	Энергетика
ETLN	Эталон	Недвижимость
FEES	ФСК Россети	Электроэнергетика
FESH	ДВМП	Машиностроение
FLOT	Совкомфлот	Транспорт
GEMC	United medical group	Здравоохранение
GLTR	Глобалтранс	Транспорт
HHRU	HeadHunter	Потребительские товары и услуги
LENT	Лента	Потребительские товары и услуги
LSRG	ЛСР	Недвижимость
MDMG	Мать и дитя	Здравоохранение
MRKC	Россети Центр	Электроэнергетика
MRKP	Россети Центр и Приволжье	Электроэнергетика
MSNG	Мосэнерго	Электроэнергетика
MSRS	Россети Московский регион	Электроэнергетика
MTLR	Мечел	Сырьевая промышленность
MTLRP	Мечел привилегированные	Сырьевая промышленность
MVID	Мвидео	Потребительские товары и услуги
OGKB	Вторая генерирующая компания оптового рынка электроэнергии	Электроэнергетика
POLY	Полиметал	Сырьевая промышленность
POSI	Positive Technologies	Информационные технологии
RASP	Распадская	Сырьевая промышленность
RENI	Ренессанс Страхование	Финансовый сектор
RNFT	РуссНефть	Энергетика
RTKM	Ростелеком	Телекоммуникации
RTKMP	Ростелеком привилегированные	Телекоммуникации
SELG	Селигдар	Сырьевая промышленность
SGZH	Сегежа	Сырьевая промышленность
SMLT	ГК Самолет	Недвижимость
TGKA	ТГК-1	Электроэнергетика
UPRO	Юнипро	Электроэнергетика
VKCO	ВК	Информационные технологии

Итоговый датасет состоит из 41 столбца, каждый представляет итоговую дневную цену закрытия акции начиная с 1 января 2015 года по 1 апреля 2024 года.

Прежде чем начать обучение нейронных сетей, для собранных данных нужно было провести предобработку данных.

Предобработка данных является крайне важным этапом подготовки информации перед использованием в РНС. В зависимости от конкретной задачи и характеристик данных, предобработка может включать в себя несколько шагов:

1. Обработка пропущенных данных.

В исходных данных могут встречаться пропущенные значения. Их можно заполнить, используя различные методы, такие как линейная интерполяция, заполнение средним значением или прогнозирование недостающих значений на основе имеющихся данных.

2. Удаление выбросов.

Так как выбросы могут искажать обучение модели, их удаление может увеличить показатели метрик. Для удаления выбросов может быть использовано применение фильтров, таких как медианный фильтр или фильтр Хэмпеля, для удаления аномальных значений.

3. Нормализация данных.

Ключевой метод, который можно описать как процесс приведения значений признаков к одному масштабу, обычно в диапазоне от 0 до 1 или -1 до 1. Нормализация помогает стабилизировать обучение модели и ускорить сходимость алгоритма оптимизации.

4. Учет сезонности.

Так как временные ряды могут содержать сезонные колебания, которые могут быть важны для прогнозирования, поэтому необходимо это учитывать. Учет сезонности может включать в себя применение методов сезонной

декомпозиции или создание дополнительных признаков, отражающих сезонные закономерности.

5. Выбор длины и шага окна.

При использовании временных рядов в качестве входных данных для рекуррентных нейронных сетей важно определить длину и шаг окна. От этого может зависеть итоговая точность модели. Длина окна определяет количество прошлых временных шагов, которые используются для прогнозирования будущих значений. А шаг окна определяет, на сколько временных шагов смещается каждое окно при обработке данных.

Одной из особенностью рекуррентных нейронных сетей является то, что они не могут обрабатывать временные ряды, содержащие пропуски. В нашем датасете у некоторых акций имелись пропуски, но меньше 3 %. Это связано с тем, что торги некоторыми акциями приостанавливались, например из-за редомиляции (перенос юридического адреса компании из одной юрисдикции в другую без прекращения работы в первоначальной стран) или других причин. Данные пропуски было решено заполнить с помощью метода линейной интерполяции. При использовании данного метода отсутствующие значения в последовательности данных восстанавливаются путем аппроксимации линейной функцией между двумя соседними известными точками.

Если рассматривать принцип работы метода линейной интерполяции более подробно, то он заключается в следующем: для каждого пропущенного значения вычисляется линейное значение, основанное на двух соседних точках, имеющих известные значения (в нашем случае это две ближайшие цены акция до пропуска и после). Это значение находится путем экстраполяции линии между ближайшими известными точками и, таким образом, заполняет пробелы в данных.

К преимуществам использования линейной интерполяции можно отнести простоту реализации и небольшое количество вычислений, что делает ее привлекательным методом для заполнения пропущенных значений

в больших наборах данных. Однако, и этот метод не является идеальным, так как при применении данного метода, особенно в случае, если пропущенные значения не обладают линейной зависимостью от ближайших известных точек, линейная интерполяция может привести к искажению данных.

Далее более подробно распишем какие инструменты для каждого из этапов использовались.

Для удаления выбросов из временных рядов использовался алгоритм Hampel Filter из библиотеки sktime. Этот алгоритм основан на скользящем окне, в пределах которого вычисляются медиана и среднее абсолютное отклонение. Затем каждое наблюдение в окне сравнивается с медианой, и, если разница превышает заданный порог, оно считается выбросом и заменяется на медиану.

Важным аспектом предобработки данных для рекуррентных нейронных сетей также является учет сезонности. Неоднозначность с точностью, с которой РНС могут моделировать сезонные колебания, создает некоторые вызовы. Некоторые исследователи считают, что при достаточной длине входного окна РНС могут моделировать сезонность [3], но другие указывают на то, что это не всегда достигается с высокой точностью [4]. В контексте анализа временных рядов сезонные колебания могут быть связаны с периодичными событиями, например такими как даты выплаты дивидендов в финансовых данных.

Для удаления сезонности в нашей работе мы использовали Deseasonalizer из библиотеки sktime, который применяет метод сезонной декомпозиции временного ряда с использованием скользящего среднего.

Еще одним этапом предобработки данных для использования в рекуррентных нейронных сетях является нормализация. Функции активации, такие как сигмоида и гиперболический тангенс, имеют область насыщения, поэтому входные данные должны быть нормализованы. В нашей работе мы использовали метод нормализации, который приводит все значения

временного ряда к диапазону от 0 до 1 с помощью функции MinMaxScaler из библиотеки sklearn. Данный метод использует формулу (1):

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (1)$$

Стабилизация дисперсии ряда — это способ предобработки данных при работе с временными рядами, особенно когда дисперсия меняется со временем. Нестабильная дисперсия может привести к недооценке или переоценке значимости временных изменений в ряде и, как следствие, к искаженным результатам моделирования или прогнозирования [6].

Существует несколько методов стабилизации дисперсии временного ряда, рассмотрим часть из них:

1. Логарифмическое преобразование.

В основе метода лежит применение логарифмической функции к каждому значению ряда. При работе с рядами, имеющими экспоненциальный рост или дисперсию, увеличивающуюся с уровнем, часто используется логарифмическое преобразование, чтобы сделать данные более сглаженными. Этот метод использовался в работе в дальнейшем.

2. Квадратный корень или корень n-ой степени.

Эти методы похожи на предыдущий, но применяются в данном случае в роли логарифма выступают корни числа, например квадратный или третьей степени. Они могут быть более подходящими для рядов с отрицательными значениями или более сложными динамиками изменения дисперсии.

3. Преобразование Бокса-Кокса.

С помощью данного обобщенного преобразования, которое включает в себя параметр λ, можно выбирать оптимальное преобразование в зависимости от структуры данных. Преобразование Бокса-Кокса является эффективным методом при работе с рядами, у которых дисперсия меняется в

зависимости от времени. И этот способ также использовался в работе в качестве базового преобразования.

4. Преобразование Джонсона.

Этот метод является расширением преобразования Бокса-Кокса и позволяет моделировать более широкий спектр форм распределения, включая симметричные и асимметричные распределения.

Выбор метода стабилизации дисперсии зависит от структуры данных и характеристик временного ряда. Целью этих методов является приведение дисперсии ряда к постоянному уровню, что позволяет более точно моделировать и прогнозировать его поведение.

Далее рассмотрим влияние каждого метода на результат прогнозирования нейронной сети и определим оптимальную комбинацию используемых методов предобработки данных.

2.3 Подбор и обучение рекуррентных нейронных сетей

Для подбора и обучения рекуррентных нейронных сетей (RNN, PHC) был проведен сравнительный анализ между простыми LSTM, GRU и Bidirectional LSTM моделями, рассмотренных в параграфе 1.3. После тщательного анализа результатов выбор пал на Bidirectional LSTM благодаря его способности улавливать как прошлую, так и будущую информацию при обработке последовательностей данных. Это позволяет модели лучше учитывать контекст при прогнозировании и классификации.

Для формирования последовательностей данных использовался метод скользящего окна. Этот метод позволяет создавать последовательности из временных данных, которые будут использоваться для обучения модели. Алгоритм его работы состоит в следующем: сначала определяется размер окна (временной шаг), затем окно последовательно перемещается по временному ряду, формируя последовательные фрагменты данных, которые

будут использоваться для обучения модели. Наглядно метод представлен на рисунке 8.

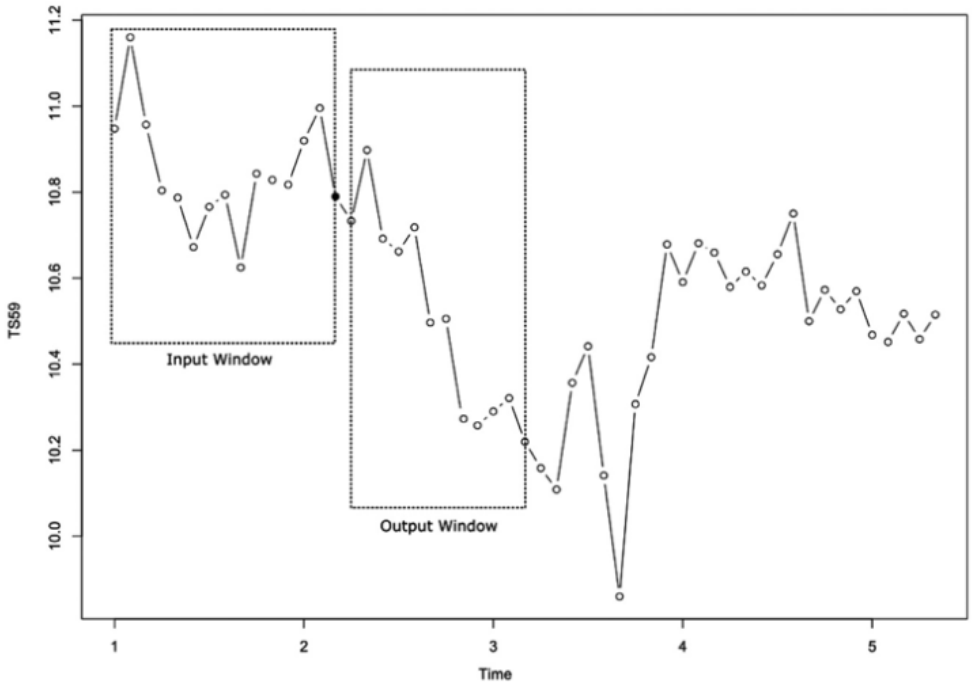

Рис.8. Метод скользящего окна

Для построения и обучения рекуррентной нейронной сети (RNN) с использованием архитектуры, представленной ниже, мы используем модель Sequential из библиотеки Keras. Эта архитектура состоит из нескольких слоев LSTM и полносвязного слоя Dense для предсказания выходной последовательности. По замыслу, для каждого временного ряда акции мы будем обучать и сохранять отдельную модель. Такой подход, построение локальных моделей вместо одной глобальной, рассматривался в статье [10], и показал хорошие показатели.

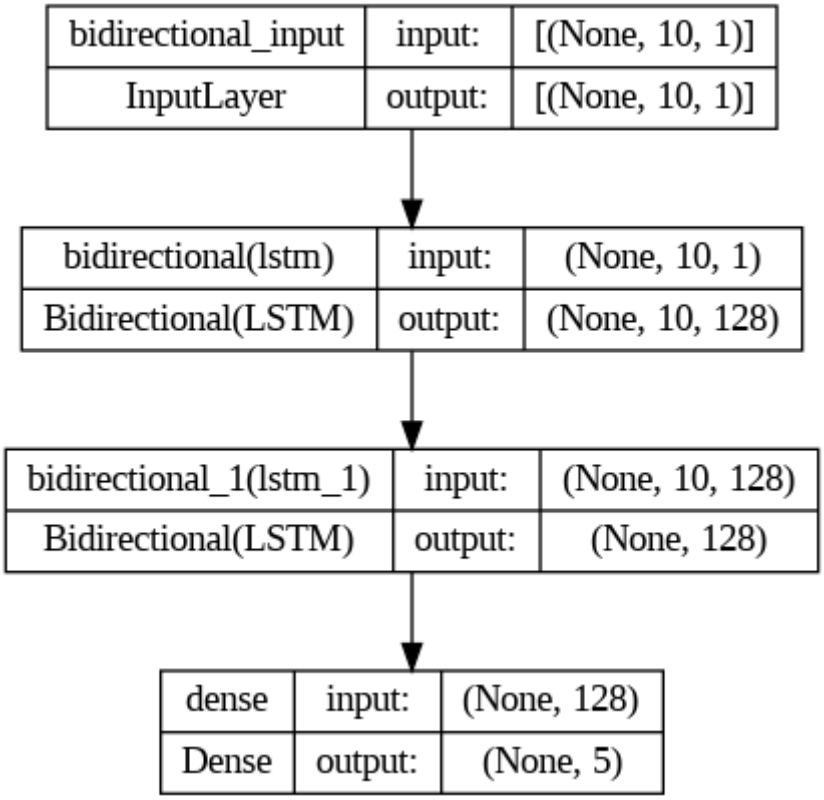

Рис.9. Структура нейронной сети

Эта модель состоит из следующих компонентов:

1. Bidirectional LSTM слои: В этой архитектуре используются два слоя Bidirectional LSTM для захвата как прямых, так и обратных последовательностей данных. Это позволяет модели лучше улавливать зависимости в данных. Оба слоя имеют по 64 блока LSTM и инициализируются начальными весами с помощью заданной константы.

2. Полносвязный Dense слой: после LSTM слоев следует полносвязный слой с функцией активации ReLU для предсказания выходной последовательности. Размер этого слоя равен количеству временных шагов в выходной последовательности (n_steps_out).

3. Компиляция модели: для компиляции модели мы используем оптимизатор Adam с коэффициентом скорости обучения 1e-3 и функцию потерь mean squared error (среднеквадратичная ошибка, формула (2), так как решается задача регрессии.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \overline{y_i})^2 \quad (2)$$

где y_i – истинное значение, $\overline{y_i}$ – предсказанное значение

Для обучения и тестирования модели была использована метод кросс-валидации KFold, который позволяет оценить качество модели на различных выборках данных и избежать переобучения. KFold кросс-валидация случайным образом разбивает данные на k непересекающихся блоков примерно одинакового размера. Поочередно каждый блок рассматривается как тестовая выборка, а остальные k-1 блоков — как обучающая выборка. Также помимо KFold был опробован другой вариант кросс-валидации, использующийся в анализе временных рядов – TimeSeriesSplit. В этом подходе начинаем обучать модель на небольшом отрезке временного ряда, от начала до некоторого t, делаем прогноз на $t+n$ шагов вперед и считаем ошибку. Далее расширяем обучающую выборку до $t+n$ значения и прогнозируем с $t+n$ до $t+2*n$, так продолжаем двигать тестовый отрезок ряда до тех пор, пока не упрёмся в последнее доступное наблюдение. В итоге получим столько фолдов, сколько n уместится в промежуток между изначальным обучающим отрезком и всей длиной ряда. Пример работы TimeSeriesSplit представлен на рис.10.

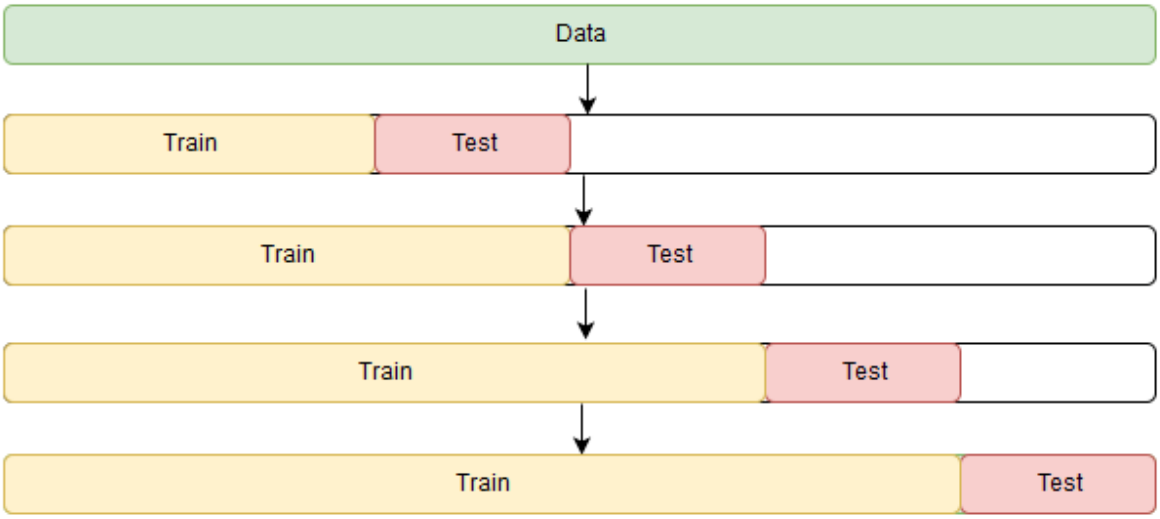

Рис.10. Разбиение данных с помощью TimeSeriesSplit.

Из-за различных масштабов временных рядов мы использовали среднюю абсолютную процентную ошибку (MAPE) в качестве целевой метрики, определенной по формуле (3). Также мы учитывали симметричную среднюю абсолютную процентную ошибку (SMAPE), представленную формулой (4), которая широко используется в исследованиях. Значения обеих метрик рассчитывались на каждой итерации кросс-валидации для каждого временного ряда, как на валидационной, так и на тестовой выборках.

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \overline{y_i}}{y_i}\right| \quad (3)$$

$$SMAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - \overline{y_i}|}{(|y_i| + |\overline{y_i}|)/2} \quad (4)$$

где y_i – истинное значение, $\overline{y_i}$ – предсказанное значение.

Выбор размера входного и выходного окна, а также количества скрытых слоев и нейронов в слое происходил на основе экспериментов с различными значениями. Была проведена серия обучений, изменяя эти параметры, и проводя оценку качества модели на валидационной выборке. Исходя из результатов, были выбраны оптимальные значения параметров, которые давали наилучшее качество прогнозирования. В итоге размер входного окна длиной 10 дней и размер выходного окна длиной 5 дней были определены как оптимальные, модели показывали в среднем ошибку в 4%. Увеличение размера входного окна не улучшило метрики, так как акции, выбранный в качестве объекта прогнозирования имеют высокую волатильность, и могут за несколько дней измениться на десятки процентов в разные стороны. Для того чтобы не упустить эти всплески, модель должна быстро реагировать на эти изменения, чего не было при использовании более длинных временных рядов. К тому же чем больше размер входного окна, тем меньше будет выборка для обучения, что также негативно влияет на показатели метрик.

Подбор гиперпараметров был осуществлен с помощью библиотеки hyport. Для этого было разработано пространство поиска, включающее в себя следующие параметры: количество временных шагов для входных данных (n_steps_in), количество временных шагов для выходных данных (n_steps_out), количество скрытых слоев (n_layers), количество нейронов в скрытых слоях (n_units), функция активации (activation), количество эпох обучения (epochs) и размер пакета данных (batch_size).

Процесс оптимизации осуществлялся с использованием алгоритма Tree-structured Parzen Estimator (TPE), который осуществляет поиск оптимальных параметров, исследуя итеративно различные комбинации значений гиперпараметров. Каждая итерация обучения модели представляла собой уникальный этап, в ходе которого модель обучалась на обучающей выборке и проверялась на валидационной выборке.

По итогу был получен наилучший набор гиперпараметров, обеспечивающий оптимальную производительность модели. Полная таблица всех гиперпараметров и их значений представлена ниже.

Таблица 2. Гиперпараметры нейронной сети

Количество скрытых слоев:	2
Количество нейронов в слоях:	64-64
Функция активации:	ReLU
Функция потерь:	MSE
Кол-во эпох:	50
Шаг сходимости:	0.0001
Оптимизатор:	Adam
Размер пакета данных	32

Как мы рассматривали в прошлом параграфе, существует множество вариантов предобработки временных рядов. Одной из задач стояла в определении влияния каждого этапа предобработки на итоговый результат и

поиск наиболее подходящей комбинации методов. На рисунке 11 представлена схема эксперимента.

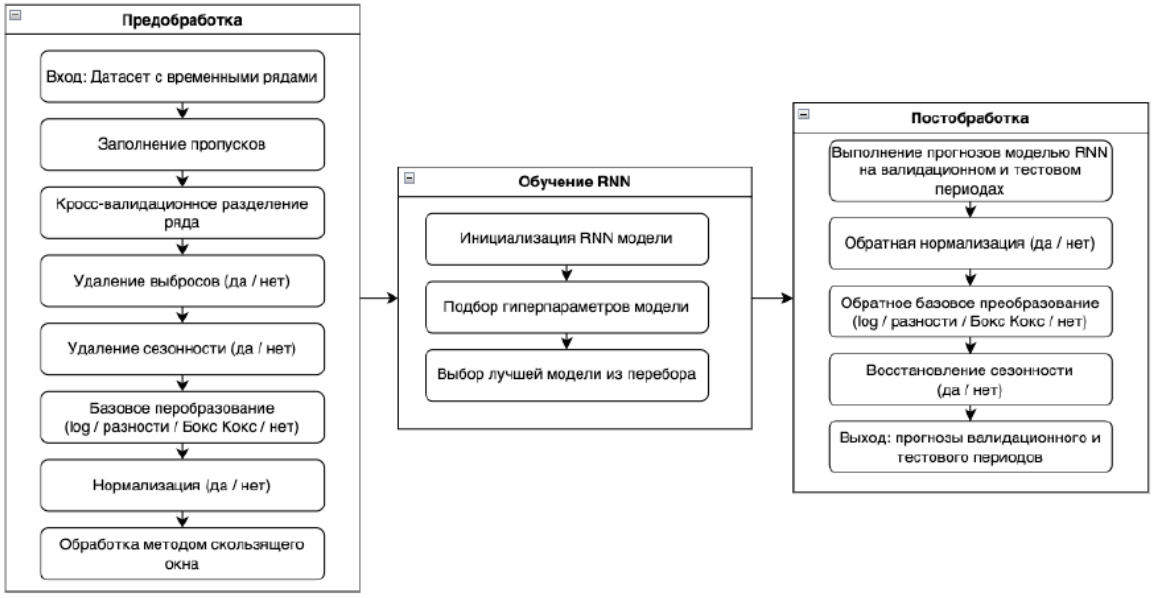

Рис.11. Схема работа с данными.

Для каждого временного ряда из анализируемой выборки в первую очередь проводилась процедура удаления или заполнения пропусков. Затем ряд разделялся в соответствии с алгоритмом кросс-валидации и нарезался на последовательности с помощью метода скользящего окна. После этого выполнялись четыре этапа предварительной обработки:

- Удаление выбросов
- Удаление сезонности
- Базовое преобразование
- Нормализация

Под базовым преобразованием подразумевается применение методов, таких как логарифмирование, преобразование Бокса-Кокса или вычисление первой разности. Эти методы применялись в зависимости от характера данных, и они могли не выполняться на указанных этапах предварительной обработки. По итогу получилось 32 комбинации методов предобработки. Из них в таблице 3 представлены 10 лучших вариантов по средним показателям

метрик MAPE и SMAPE по 7 акциям, показавшим наихудшие показатели без предобработки данных.

Таблица 3. 10 лучших комбинаций предобработки

Удаление выбросов	**Удаление сезонности**	**Базовое преобразование**	**Нормализация**	**SMAPE**	**MAPE**
Да	Да	log	Да	6.02	5.86
Нет	Нет	log	Да	6.51	6.27
Да	Да	boxcox	Да	6.77	6.60
Да	Нет	boxcox	Да	6.86	6.65
Нет	Нет	log	Да	7.12	6.71
Нет	Нет	boxcox	Да	7.24	7.05
Нет	Да	log	Да	7.71	7.34
Нет	Нет	Нет	Да	8.17	7.59
Да	Нет	Нет	Да	8.26	7.68
Да	Да	Нет	Да	8.48	7.91

Проанализировав данные таблицы, можно заметить, что во всех лучших вариантах использовалась нормализация, а также логарифмическое преобразования или Бокса-Кокса. Вкупе эти два подхода давали хорошие результаты на тестовой выборке. Удаление выбросов и сезонности в целом может улучшить показатели, но не во всех случаях. При подходе, когда не использовались никакие методы предобработки, средний показатель MAPE по всем акциям был 28.9, тогда как средний показатель при самой лучшей комбинации предобработки из таблицы 3 был 4.19, то есть результат улучшился почти в 7 раз. На рисунке 12 продемонстрированы результаты для каждого временного ряда до предобработки данных и с предобработкой, в которой использовалась удаление выбросов, удаление сезонности, логарифмическое преобразование и нормализация.

Рис.12. Сравнение результатов с предобработкой данных и без нее

Как можно заметить на гистограмме, для части временных рядов результат не изменился, тогда как для другой части нейронные сети показывали плохие результаты. При использовании методов предобработки нейронные сети проблемных временных рядов смогли адаптироваться и улучшить свои результаты в несколько раз. Снижение процента ошибки наблюдается, например у акции SMLT, было 94 % - стало 1.98 %

Таким образом, была определена оптимальная комбинация методов предобработки, и были обучены и сохранены 41 модель, а также преобразователи данных для каждой из них.

2.4 Разработка алгоритма формирования инвестиционного портфеля

Для решения задачи распределения активов использовались алгоритмы обучения с подкреплением.

Обучение с подкреплением (Reinforcement Learning) — это область машинного обучения, которая фокусируется на обучении программных агентов принимать действия в среде для максимизации некоторой числовой

ценности (вознаграждения), представляющей долгосрочную цель. Схема этого процесса показана на рисунке 13.

Рис.13. Схема обучения с подкреплением.

Агент и окружающая среда являются ключевыми компонентами обучения с подкреплением. Среда — это сущность, с которой агент взаимодействует. На каждом шагу взаимодействия агент наблюдает за состоянием мира и принимает решение о следующем действии. Среда изменяется под воздействием агента (она также может изменяться самостоятельно).

Агент получает вознаграждения от окружающей среды — число, которое указывает на то, насколько хорошим или плохим является состояние мира после выполнения действия.

Цель алгоритмов обучения с подкреплением состоит в том, чтобы научить агента эффективно взаимодействовать с окружающей средой, максимально увеличивая своё кумулятивное вознаграждение, называемое возвратом.

В данном примере агентом является компьютер, который наблюдает за показателями финансового рынка (состояниями). Финансовый рынок выступает в роли окружающей среды. Действия, которые агент может

предпринять, включают покупку, удержание и продажу. Агенту необходимо научиться выбирать правильные действия и время их выполнения (стратегия).

Он учится определять, какие сделки хорошие, а какие — плохие, с помощью вознаграждений. Цель обучения агента с подкреплением может меняться в зависимости от задачи — одна из них, например, научиться торговать на финансовых рынках, избегая убытков. Важно отметить, что нашей целью не является заработать как можно больше денег.

Для реализации среды был создан отдельный класс PortfolioEnv на языке Python. Во многих работах при решении задачи распределения активов в инвестиционном портфеле с помощью обучении алгоритмов с подкреплением в качестве среды используется датасет с историческими данными цен акций [13]. В этой работе было решено использовать датасет из предсказанных цен с помощью нейросетей для каждой акции, описанных в параграфе 2.3. Данный сгенерированный набор данных создавался используемым ранее методом скользящего окна. Используя 10 исторических цен, прогнозировались цены на 5 дней, после чего окно сдвигалось на 5 дней вперед и процесс повторялся. По итогу объединив все отрезки с предсказанными данными получался датасет предсказанных. При инициализации объекта класса он передавался в качестве параметра, а также устанавливались начальный баланс в размере 1.000.000 у.е. и комиссию за торговлю в размере 0,04% (сумма средней комиссии биржи и брокера за операцию). Гипотеза состоит в том, что имея приблизительно возможные цены акций, алгоритм сможет определить оптимальное распределение активов.

Конструктор инициирует основные переменные и задает пространство действий и наблюдений для агента. Пространство действий определяет возможные действия агента (покупка, продажа или удержание активов), а пространство наблюдений включает текущие состояния активов и свободных денег. Также в классе был реализован метод reset, который сбрасывает

состояние среды до начального (обнуление шагов симуляции, инициализация портфеля и денежных средств). Еще один метод get_state возвращает текущее состояние среды, которое включает в себя количество свободных денег на данном временном шаге, портфель активов (массив количества той или иной акции) и предсказанных цен на момент времени t.

Ключевым методом является step, который выполняет один шаг в симуляции, в результате которого агент совершает действия (покупка, продажа, удержание) в рамках установленной стратегии. Согласно ей, сначала все акции, помеченные агентом как «к продаже», продавались до нуля. Затем нужные акции докупались на одинаковую сумму в размере 2,5 % от текущих свободных денежных средств. После чего среда обновляет состояние в ответ на действие агента, пересчитывается стоимость портфеля, его доходность в процентном сравнении с предыдущей. Так как главной целью является формирование постоянно растущего портфеля, а не максимизация прибыли, процентная доходность портфеля по сравнению с предыдущей и было выбрано в качестве вознаграждения агенту. Цель агента — максимизировать кумулятивное вознаграждение, т.е. доходность портфеля. На основе полученного вознаграждения агент корректирует свою стратегию, улучшая выбор действий.

Одним из главным опасением являлось, то, как отреагирует себя подход распределения активов на предсказанных ценах на реальных данных. Для выяснения этого вопроса, все действия агента дублировались на реальном наборе данных, симулируя то, как отреагировал бы сформированный портфель в реальных условиях и в итоге сравнивалась доходность виртуального портфеля и настоящего. На рисунке 12 продемонстрированы результаты сравнения.

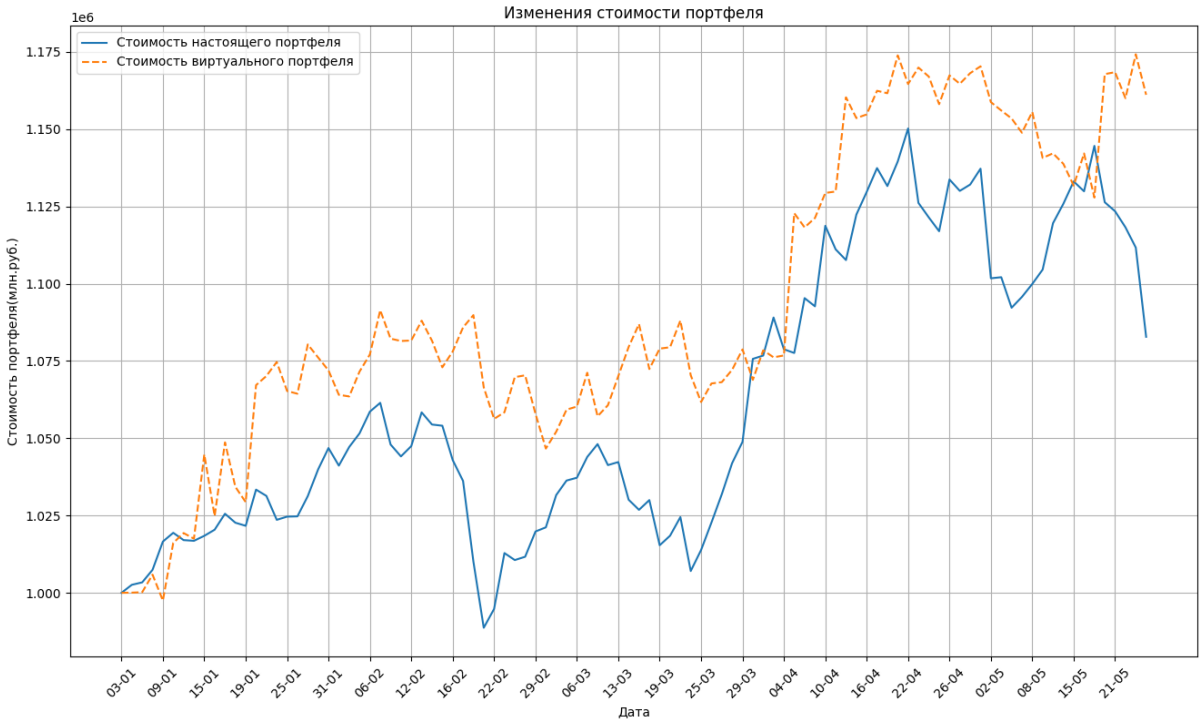

Рис.14. Сравнение стоимости портфелей.

Как можно видеть на рисунке 14, график стоимости портфеля, который бы собирался в реальных условиях, хоть и отстает от виртуального, но соблюдает его основные тренды движения с небольшим временным лагом. Это говорит, о том, что агент научился эффективно распределять активы на виртуальном пространстве наблюдений, а довольно высокая точность прогнозирования цен акций дает возможность следовать стратегии.

Отдельной задачей являлась выбор агента для этой задачи. Алгоритмов для обучения с подкреплением существует достаточное количество. Для решения финансовых задач часто используются алгоритмы из класса Актор-Критик [14]. Эти алгоритмы сочетают в себе идеи из политик-градиентных методов (Policy Gradient) и методов оценки ценности (Value-Based Methods), что позволяет им эффективно обучаться в сложных средах [15].

Такие алгоритмы состоят из двух компонентов: актор и критик. Актор представляет собой нейронную сеть, которая аппроксимирует политику (π). Политика определяет, какое действие должно быть предпринято в каждом состоянии. Актор принимает на вход текущее состояние и возвращает распределение вероятностей для всех возможных действий. Цель актора -

максимизировать ожидаемое вознаграждение, следуя политике, которая направляет выбор действий. Критик представляет собой нейронную сеть, которая аппроксимирует функцию ценности (V или Q). Функция ценности оценивает "качество" текущего состояния или состояния-действия комбинации в терминах ожидаемого будущего вознаграждения. Критик используется для уменьшения вариативности градиентов, обновляя оценку на основе разницы между ожидаемым и фактическим вознаграждением. На рисунке 15 изображена схема работа алгоритма

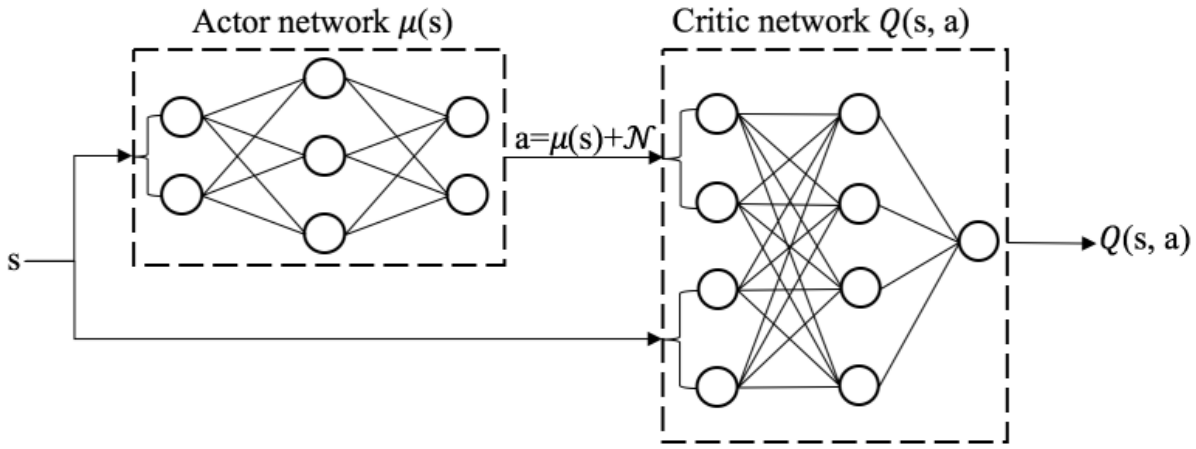

Рис.15. Архитектура Актор-Критик.

Принцип работы алгоритмов Актор-Критик

1. Вычисление действия:

Актор, используя текущую политику, выбирает действие на основе текущего состояния среды. Политика может быть детерминированной или стохастической.

2. Выполнение действия:

Выбранное действие отправляется в среду, которая отвечает новым состоянием и вознаграждением.

3. Оценка действия:

Критик оценивает текущее состояние или состояние-действие комбинацию, предоставляя оценку (ценность) ожидаемого будущего вознаграждения.

4. Обновление критика:

Ошибка TD (разница между фактическим вознаграждением плюс оценка следующего состояния и текущей оценкой) вычисляется и используется для обновления параметров критика, что улучшает оценку состояния или состояние-действие комбинации.

5. Обновление актора:

Актор обновляет свою политику на основе градиента, взвешенного по оценке критика. Градиент направлен в сторону увеличения вероятности действий, приводящих к высоким вознаграждениям.

В данной работе рассматривались четыре метода из данного класса: Advantage Actor-Critic (A2C), Deep Deterministic Policy Gradient (DDPG), Proximal Policy Optimization (PPO), Twin Delayed Deep Deterministic Policy Gradient (TD3). После обучения на тестовых данных результаты этих моделей проверялись на тестовом датасете. Результаты показаны на рисунке 16.

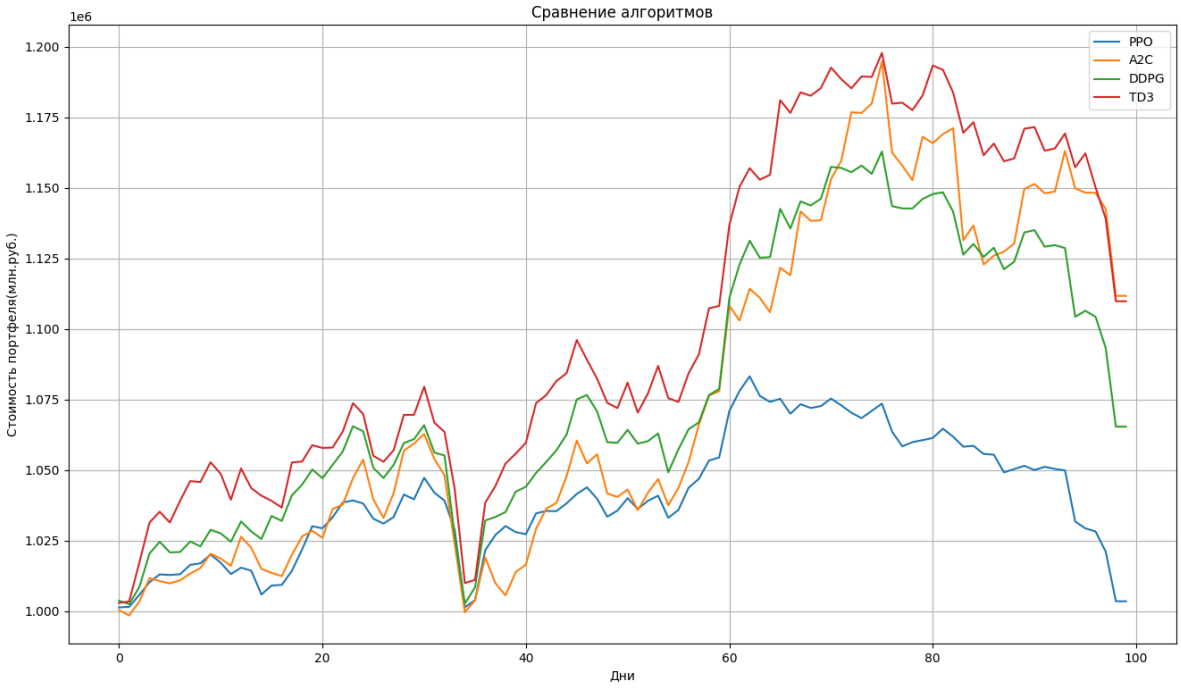

Рис.16. Сравнение алгоритмов на тестовом множестве.

Наилучший результат продемонстрировали алгоритмы A2C и TD3. Алгоритм DDPG справился с задачей на приемлемом уровне, тогда как PPO

не смог показать удовлетворительных результатов. В качестве агента было решено выбрать алгоритм Twin Delayed Deep Deterministic Policy Gradient (TD3). Обучив модель, она была сохранена для дальнейшего использования.

Глава 3. Тестирование и анализ результатов разработанной системы

3.1 Оценка эффективности инвестиционного портфеля

Для оценки эффективности сформированных алгоритмами инвестиционных портфелей, была создана среда, состоящая из 40 ежедневных цен акций, предсказанные рекуррентными нейронными сетями. Для теста использовались данные с января 2024 года по середину мая 2024. Получив результаты доходности портфелей, которые сформировали алгоритмы A2C, DDPG PPO, TD3, нужно было их сравнить с бенчмарком. В нашем случаем им выступает индекс МосБиржи компаний средней и малой капитализации (тикер MCXSM), который и состоит из акций компаний, рассмотренных в параграфе 2.2. Также в качестве примера для сравнения был создан портфель, состоящий из равных долей всех акций индекса. Доходности этих портфелей также были взяты за период тестовых данных агентов.

Для оценки результатов используются четыре показателя: итоговая стоимость портфеля, месячная доходность, месячная стандартная ошибка и коэффициент Шарпа. Итоговая стоимость портфеля отражает его стоимость в конце торгового этапа. Месячная доходность показывает прямую доходность портфеля за год. Месячная стандартная ошибка демонстрирует устойчивость нашей портфеля. Коэффициент Шарпа объединяет доходность и риск для такой оценки и рассчитывает по следующей формуле (5):

$$Sharpe\ Ratio = \frac{R_p - R_f}{\sigma_p}\ , \qquad (5)$$

где R_p— доходность портфеля, R_p— безрисковая ставка (обычно используется процентная ставка сберегательного счета за вычетом инфляции), а σ_p— стандартное отклонение доходности портфеля за рассматриваемый период времени.

В качестве начальной точки стоимости портфелей была выбрана сумма в 1.000.000 рублей. Результаты сравнения стоимостей портфелей показана на рисунке 17.

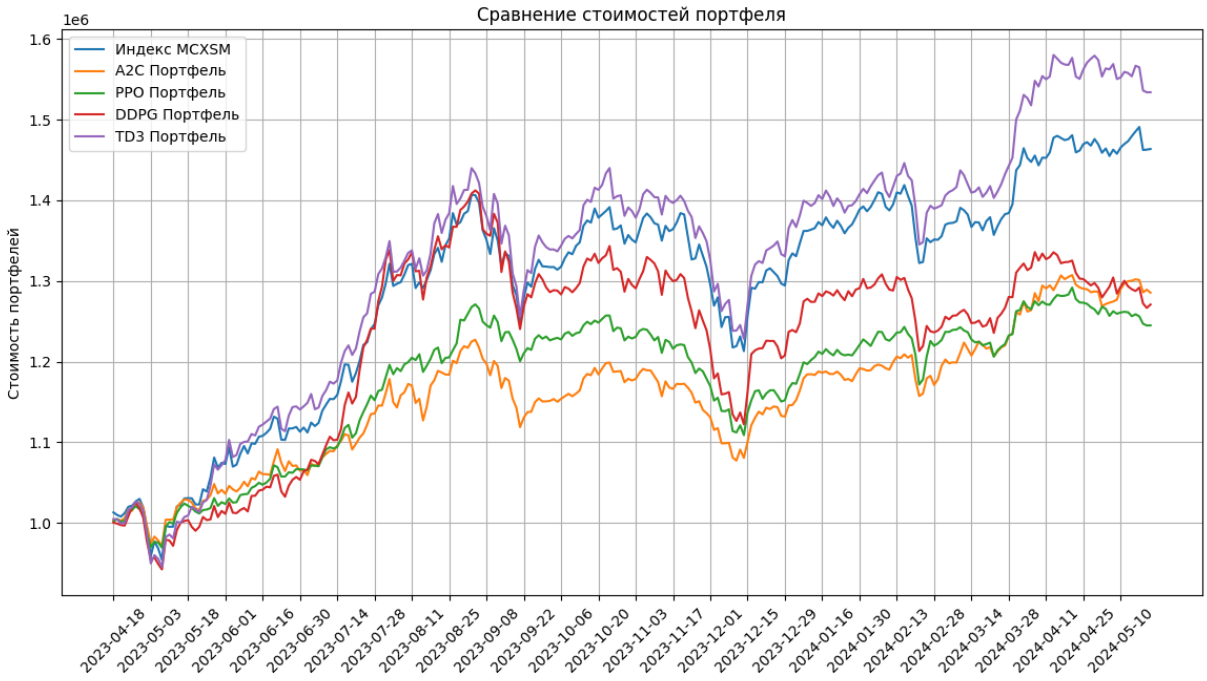

Рис.17. Сравнение доходностей портфелей.

Как можно увидеть, наилучшие результаты по сравнению с другими портфелями демонстрирует TD3. В начале года его стоимость была примерно равна стоимости других портфелей, но затем он начал превосходит их, достигая максимального значения около 1.57 миллиона к концу марта 2024 года. При этом портфель имеет умеренные колебания стоимости и на всем протяжении обыгрывает индекс.

Портфель от алгоритма TD3 также показал хорошие результаты, стабильно растет на протяжении всего периода и к апрелю 2024 года достигает второго по величине значения стоимости.

Далее идет портфель индекса, бенчмарк, которого и должны превзойти остальные портфели.

DDPG сформировал портфель, который является самым волатильным и достигает своего пика в середине августа 2023 года, что было выше всех. Однако позже его стоимость опустилась вниз, став одним из самых

убыточных. Под конец же периода портфель отрывался от двух портфелей-аутсайдеров.

Портфели, собранные алгоритмами PPO A2C, показывают наихудшие результаты среди всех представленных. Их стоимость растет очень медленно и, хотя они следуют трендам индекса и других портфелей. В целом, их стоимость остается самой низкой на протяжении всего периода.

Таблица 3. Результаты портфелей

	DDPG	TD3	A2C	Индекс	PPO
Начальная стоимость портфеля	1000000	1000000	1000000	1000000	1000000
Конечная стоимость портфеля	1270780	1534015	1285494	1463658	1244928
Ежемесячная доходность	2.88 %	4.19 %	2.88 %	3.68 %	2.3 %
Ежемесячное стандартное отклонение	8.71 %	6.74 %	4.16 %	6.52 %	4.93 %
Коэффициент Шарпа	0.32	0.61	0.58	0.55	0.45

Итоговая доходность портфеля TD3 выше итоговой доходности индекса на 4,8 %.

Таким образом, результат демонстрирует, что предлагаемая стратегия TD3 может эффективно разработать торговую стратегию, которая превосходит индекс МосБиржи компаний средней и малой капитализации.

3.2 Внедрение прогнозного алгоритма в приложение

В данном разделе рассматривается процесс интеграции прогнозного алгоритма в приложение, реализованное на платформе Streamlit. Приложение предназначено для управления инвестиционным портфелем и включает в себя функционал для загрузки данных, расчета рекомендаций и визуализации результатов.

Streamlit — это фреймворк для создания веб-приложений на Python, ориентированная на разработку приложений для анализа данных и машинного обучения. Он позволяет создавать веб-приложения с минимальными усилиями, используя простой и интуитивно понятный синтаксис Python. Для создания интерактивного приложения достаточно добавить несколько строк кода. Все компоненты, такие как графики, таблицы и формы, создаются с помощью встроенных функций Streamlit. Это значительно ускоряет процесс разработки и тестирования новых идей и гипотез. На рисунке 18 представлен начальный экран приложения.

Рис.18. Стартовый экран приложения.

Для начала работы приложения на боковой панели пользователь может выбрать диапазон дат, за который будет производиться анализ. По умолчанию установлены текущая неделя для удобства пользователя. Затем приложение загружает данные с Московской биржи, используя API Moex и

библиотеку apimoex. Загруженные данные включают в себя исторические цены акций и значения индекса. После загрузки данных они сохраняются в сессии для дальнейшего использования.

```python
# Функция для загрузки данных
@st.cache_data(ttl=3600)
def load_data(start_date, end_date):
    with requests.Session() as session:
        data = apimoex.get_index_tickers(session, index: 'MCXSM', date: '2023-12-29')
        df_tickers = pd.DataFrame(data)
        df_final = pd.DataFrame()
        for ticker in df_tickers['ticker']:
            data = apimoex.get_board_history(session, ticker, start=start_date, end=end_date)
            df = pd.DataFrame(data)
            df.set_index( keys: 'TRADEDATE', inplace=True)
            df = df.rename(columns={"CLOSE": ticker})
            df_final = pd.concat( objs: [df_final, df[ticker]], axis=1)
        df_final.index = pd.to_datetime(df_final.index)
        url = (f'https://iss.moex.com/iss/history/engines/stock/markets/index/boards/'
               f'SNDX/securities/MESMTR.xml?from={start_date_str}&till={end_date_str}')
        response = requests.get(url)
        # Проверка успешности запроса
        if response.status_code == 200:
            # Парсинг XML
            root = ET.fromstring(response.content)
            # Получение всех элементов 'row'
            rows = root.findall('.//row')
            # Извлечение значений из поля 'CLOSE'
            index_arr = [float(row.attrib['CLOSE']) for row in rows]
        else:
            print(f"Ошибка при выполнении запроса: {response.status_code}")
            index_arr = []
        return df_final, index_arr
```

Рис.19. Функция загрузки данных

Параметры для функции получают из полей на форме, сама загрузка начинается по кнопке «Загрузка данных». Для сохранения данных внутри сессии используется параметр session_state, в котором можно хранить данные в виде словаря.

```python
st.set_page_config(layout="wide", page_title="Portfolio Dashboard", page_icon="📈")
# Выбор дат
today = date.today()
start_of_week = today - timedelta(days=today.weekday())
end_of_week = start_of_week + timedelta(days=6)
start_date = st.sidebar.date_input("Начало периода", start_of_week)
end_date = st.sidebar.date_input("Конец периода", end_of_week)
start_date_str = start_date.strftime("%Y-%m-%d")
end_date_str = end_date.strftime("%Y-%m-%d")

# Кнопка для загрузки данных
if st.sidebar.button('Загрузить данные') or len(st.session_state) == 0:
    df_final, MCXSM_index = load_data(start_date_str, end_date_str)
    tickers = df_final.columns
    st.session_state['tickers'] = tickers
    predictions, prices = utils.get_dataframe(df_final)
    stocks_data = {}
    for ticker in tickers:
        stocks_data[ticker] = {'prices': prices[ticker].values, 'predicted': predictions[ticker].values,
                               'name': ticker,
                               'date_index': prices[ticker].index}
    st.session_state['stocks_data'] = stocks_data
    st.session_state['df_predicted'] = predictions
    st.session_state['df_final'] = df_final
    st.session_state['MCXSM_index'] = MCXSM_index

# Кнопка для загрузки данных
if st.sidebar.button('Загрузить данные') or len(st.session_state) == 0:
    df_final, MCXSM_index = load_data(start_date_str, end_date_str)
    tickers = df_final.columns
    st.session_state['tickers'] = tickers
    predictions, prices = utils.get_dataframe(df_final)
    stocks_data = {}
    for ticker in tickers:
        stocks_data[ticker] = {'prices': prices[ticker].values, 'predicted': predictions[ticker].values,
                               'name': ticker,
                               'date_index': prices[ticker].index}
    st.session_state['stocks_data'] = stocks_data
    st.session_state['df_predicted'] = predictions
    st.session_state['df_final'] = df_final
    st.session_state['MCXSM_index'] = MCXSM_index
```

Рис.20. Код отрисовки полей периода и кнопки «Загрузка данных»

На боковой панель есть раскрывающийся список «Мой портфель», в котором пользователь может ввести количество каждой акции, которая у него есть. Данные этого портфеля сохраняются в базе данных и далее используются для генерации рекомендаций. Код и визуальный вид элемента представлены на рисунках ниже.

```python
# Вкладка "Мой портфель"
with st.sidebar.expander("Мой портфель", expanded=False):
    portfolio_input = {}
    for stock in tickers:
        portfolio_input[stock] = st.number_input(f'{stock} акций', min_value=0, value=0)
        st.write(f'Текущая цена: ₽ {stocks_data[stock]["prices"][-1]}')
    st.session_state['portfolio'] = portfolio_input

# Обновление данных круговой диаграммы на основе ввода пользователя
new_portfolio_df = pd.DataFrame(list(portfolio_input.items()), columns=['Акция', 'Сумма'])

# Круговая диаграмма
pie_fig = px.pie(new_portfolio_df, values='Investment', names='Stock', title='Распределение активов', hole=0.3,
                 template='plotly_dark')
pie_fig.update_traces(textposition='inside', textinfo='percent+label', pull=[0.1, 0, 0, 0])

# Круговая диаграмма портфеля алгоритма
sector_pie_fig = px.pie(sector_df, values='Investment', names='Stock', title='Портфель алгоритма', hole=0.3,
                        template='plotly_dark')
sector_pie_fig.update_traces(textposition='inside', textinfo='percent+label', pull=[0.1, 0, 0, 0])
```

Рис.21. Код отображения вкладки «Мой портфель» и заполнения диаграмм

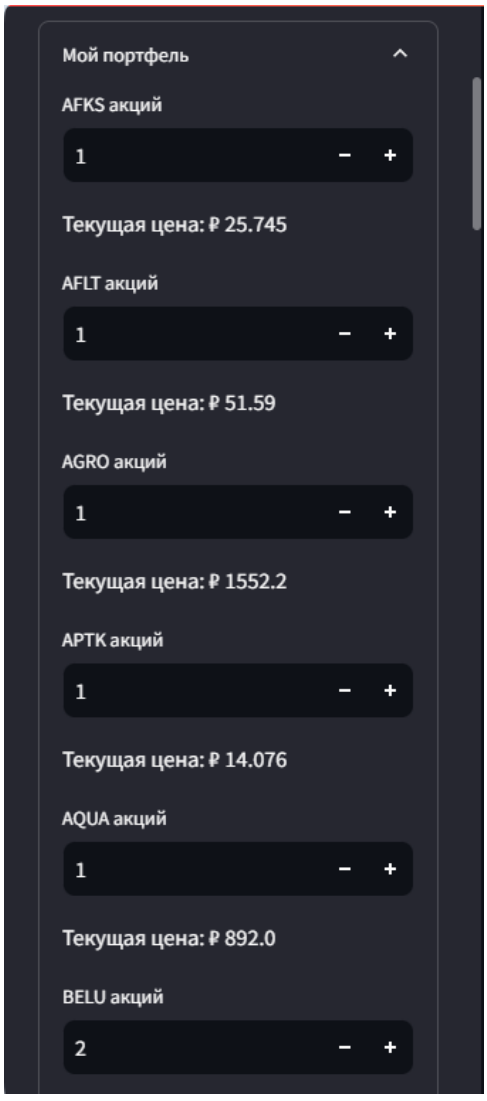

Рис.22. Вкладка «Мой портфель»

Для хранения данных пользователя и системы используются SQLAlchemy и SQLite. Это позволяет эффективно управлять данными и обеспечивает надежное их хранение. SQLAlchemy предоставляет мощный ORM (Object-Relational Mapping) слой, который упрощает работу с базой данных и интеграцию с приложением. В базе данных хранятся акции пользователя, чтобы не вводить из каждый раз, данные портфеля, сформированного прогнозным алгоритмом, запись его действий.

Для лучшей визуализации также добавлены круговые диаграммы портфелей пользователя и прогнозного алгоритма (рис.23). Левая диаграмма заполняется на основе данных, введенных пользователем во вкладке «Мой портфель», правая представляет собой данные портфеля, который предлагает рекомендательный алгоритм. Код заполнения диаграмм продемонстрирован на рисунке 23.

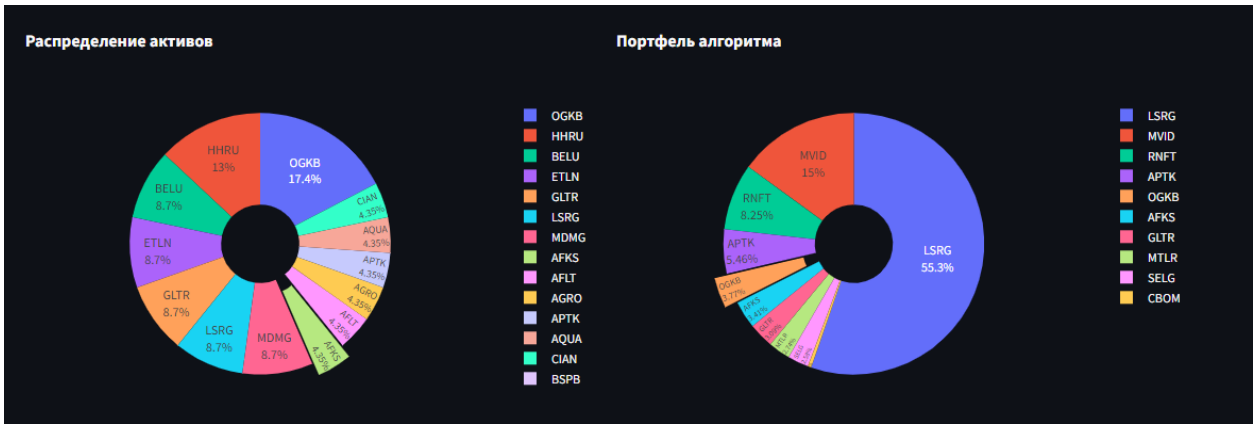

Рис.23. Визуализации распределения активов.

Основной график приложения отображает динамику стоимости портфеля и индекса, что позволяет пользователю визуально сравнить их изменения.

```python
# Создание DataFrame
df = pd.DataFrame(data)
df['Portfolio Return'] = df['Stock Price'].pct_change().fillna(0)
df['Index Return'] = df['Index'].pct_change().fillna(0)
# Накопленная доходность
df['Cumulative Portfolio Return'] = (1 + df['Portfolio Return']).cumprod() - 1
df['Cumulative Index Return'] = (1 + df['Index Return']).cumprod() - 1
# Преобразование доходности в проценты
df['Cumulative Portfolio Return Percent'] = df['Cumulative Portfolio Return'] * 100
df['Cumulative Index Return Percent'] = df['Cumulative Index Return'] * 100
sector_df = pd.DataFrame(sector_data)

# Основной график
fig = go.Figure()
fig.add_trace(
    go.Scatter(x=df['Date'], y=df['Cumulative Portfolio Return Percent'], mode='lines+markers', name='Портфель',
               line=dict(color='orange'), fill='tozeroy', fillcolor='rgba(255, 165, 0, 0.2)'))
fig.add_trace(
    go.Scatter(x=df['Date'], y=df['Cumulative Index Return Percent'], mode='lines+markers', name='Индекс MCXSM',
               line=dict(dash='dash', color='white')))

fig.update_layout(
    title='График накопленной доходности портфеля и индекса',
    xaxis=dict(title='Дата'),
    yaxis=dict(title='Накопленная доходность (%)'),
    yaxis_tickformat='%{n}',
    legend=dict(x=0.01, y=0.99)
)
```

Рис.24. Код построения основного графика.

Рис.25. Основной график.

Справа от основного графика расположен общий баланс, который включает в себя текущую стоимость портфеля и свободные денежные средства. Также подсчитывается сколько денег было изначально вложено и

какую доходность портфель имеет сейчас. Все эти параметры в последствии подставляются в код для отображения. Streamlit дает возможность использовать части кода HTML и CSS для отдельных элементов, что продемонстрировано в коде ниже (рисунок 26). На рисунке 27 показан сам элемент.

```
# Размещение элементов
col1, col2 = st.columns([3, 1])

with col1:
    st.plotly_chart(fig, use_container_width=True)

with col2:
    st.subheader("Баланс")
    st.markdown(f"""
        <div style="background-color: #333; padding: 20px; border-radius: 10px; text-align: center;">
            <h1 style="color: white; font-size: 30px;">₽ {balance}</h1>
            <p style="color: white; font-size: 18px;">Инвестировано</p>
            <p style="color: white; font-size: 22px;">₽ {investment} </p>
            <p style="color: white; font-size: 18px;">Доходность</p>
            <p style="color: green; font-size: 22px;">₽ {gain} ({gain_percent}%)</p>
            <p style="color: white; font-size: 18px;">Кэш</p>
            <p style="color: white; font-size: 22px;">₽ {cash} </p>
        </div>
    """, unsafe_allow_html=True)
```

Рис.26. Код формирования колонки баланс.

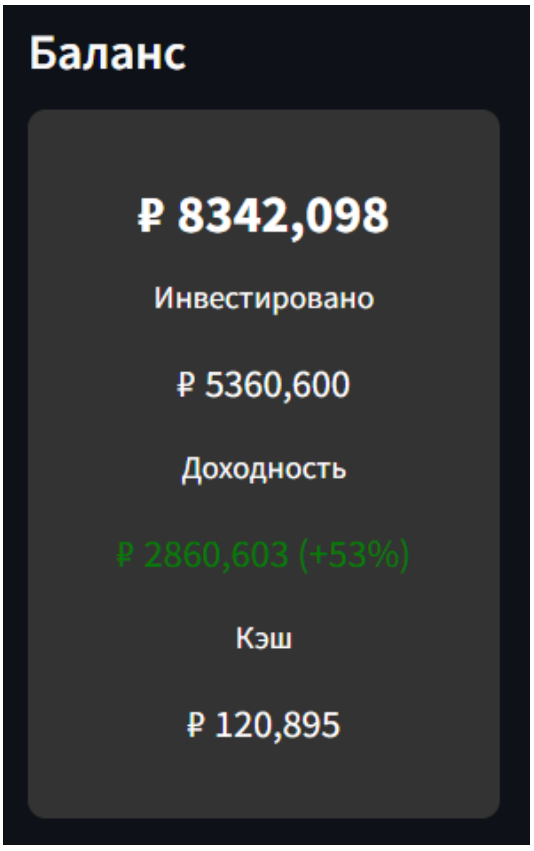

Рис.27. Элемент интерфейса «Баланс».

Для более детального анализа каждая акция имеет свой мини-график, отображающий её историю цен и прогнозируемые значения. При необходимости, есть возможность раскрыть полный график акции, где помимо текущих значений будет и прогноз цены. В зависимости от того, было ли снижение или рост за последний день, графики окрашиваются в красный и зелёный цвет соответственно. Код и сами графики представлены на рисунках 28 и 29.

```
for i in range(num_cards):
    with cols[i % 3]:
        stock = stock_cards[i]
        price_data = stocks_data[stock]['prices']
        predicted_data = stocks_data[stock]['predicted']
        name = stocks_data[stock]['name']

        # Процентное изменение цены
        change_percent = ((price_data[-1] - price_data[-2]) / price_data[-2]) * 100
        color = 'green' if change_percent >= 0 else 'red'

        # Создание мини-графика
        mini_fig = go.Figure()
        mini_fig.add_trace(go.Scatter(
            x=list(range(len(price_data))),
            y=price_data,
            mode='lines',
            line=dict(color=color),
            fill='tozeroy',
            fillcolor=f'rgba(0, 255, 0, 0.2)' if color == 'green' else f'rgba(255, 0, 0, 0.2)'
        ))
        mini_fig.update_layout(
            margin=dict(l=0, r=0, t=30, b=0),
            height=150,
            xaxis=dict(visible=False),
            yaxis=dict(range=[min(price_data), max(price_data)]),
            showlegend=False
        )
```

Рис.28. Код мини-графиков.

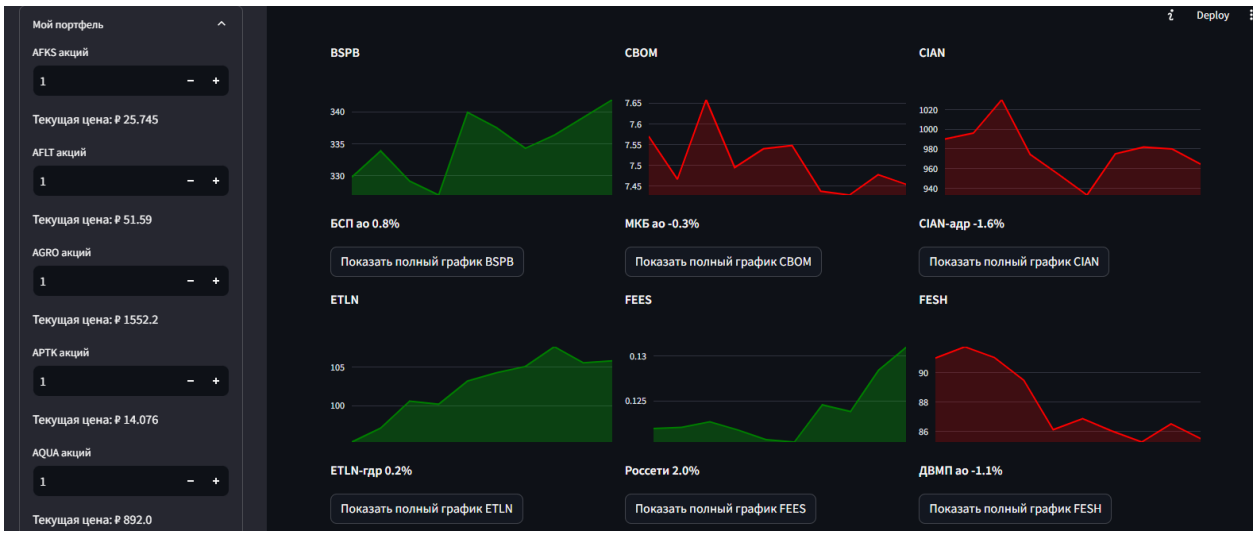

Рис.29. Мини-графики акций.

Прогнозный график строится с помощью обученных нейронных сетей, которые мы рассматривали в параграфе 2.3. При запуске приложения программа берет последние 10 торговых дней и передает их моделям, которые делают прогноз на ближайшие 5 торговых дней. Так как для корректной работы нейронных сетей помимо самих моделей требуется в должном виде подготовить и обработать данные, согласно выбранной стратегии в параграфе 2.3. Для этого загружаются сохраненные ранее для каждой акции преобразователи данных. Чтобы перегружать код приложения, все вспомогательные функции были выделены в отдельный файл utils.py.

В функции get_dataframe, загруженные раннее данные с МосБиржи разделяются по акциям, потом загружаются сохраненные преобразователи данных для каждой отдельной акции, такие как scaler, log transformer и deseanosinizer. Далее эти данные преобразуются и нарезаются в последовательности для LSTM, после чего загруженная нейронная сеть делает прогноз. Полученные данные возвращаются в приложение и отображаются на полном графике акции.

```python
def get_dataframe(data):
    all_data = {}
    for column in data.columns:
        ticker_data = data[column].dropna()
        test_data, test_index = ticker_data.values, ticker_data.index
        all_data[column] = test_data, test_index
    model_folder = "model_2"
    models, transformers_data = {}, {}
    for stock_symbol in os.listdir(model_folder):
        model_path = os.path.join(model_folder, stock_symbol, f"{stock_symbol}.keras")
        path = os.path.join(model_folder, stock_symbol)
        models[stock_symbol] = keras.saving.load_model(model_path)
        transformers_data[stock_symbol] = {'hampel': joblib.load(path + rf"\hampel_{stock_symbol}"),
                                           'deseasonalizer': joblib.load(path + rf"\deseasonalizer_{stock_symbol}"),
                                           'transformation': joblib.load(path + rf"\transformation_{stock_symbol}"),
                                           'scaler': joblib.load(path + rf"\scaler_{stock_symbol}")}
    predictions, true_values = {}, {}
    for stock_symbol, model in models.items():
        transformers = transformers_data[stock_symbol]
        test_data, test_index = all_data[stock_symbol]
        X = prepare_data_with_preprocessing(test_data, transformers)
        y_pred = model.predict(X)
        y_pred = transformers['scaler'].inverse_transform(y_pred.reshape(-1, 1)).reshape(y_pred.shape)
        y_pred = transformers['transformation'].inverse_transform(y_pred.reshape(-1, 1)).reshape(y_pred.shape)
        y_pred = transformers['deseasonalizer'].inverse_transform(y_pred.reshape(-1, 1)).reshape(y_pred.shape)
        predictions[stock_symbol] = y_pred.reshape(-1)
        true_values[stock_symbol] = test_data
    df = pd.DataFrame(predictions)
    return df
```

Рис.30. Код формирования предсказанных цен

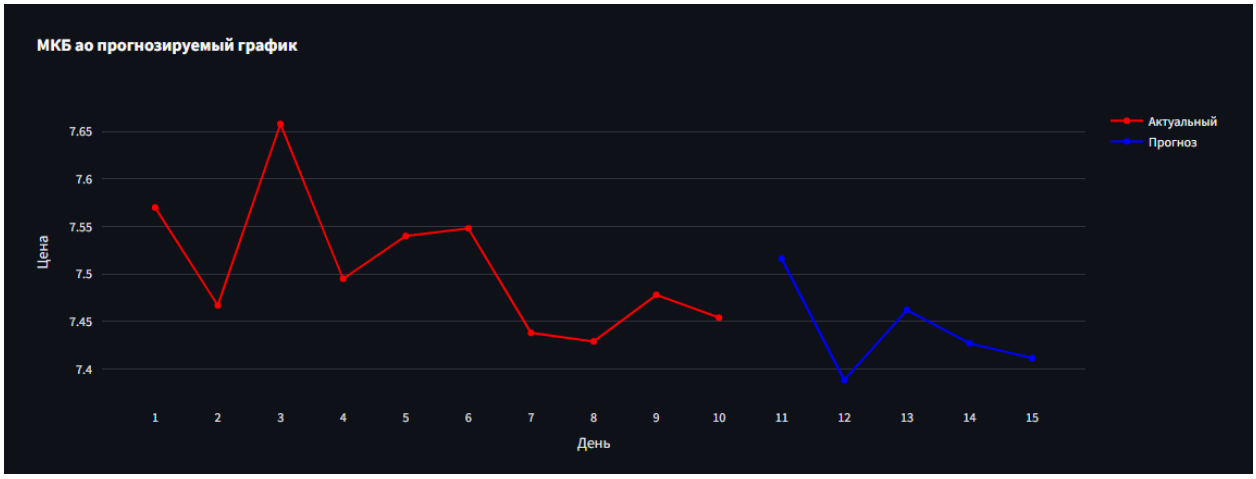

Рис.31. Раскрытый прогнозируемый график акции.

Пользователь также имеет возможность получить рекомендации по составу портфеля, нажав на соответствующую кнопку на боковой панели. Приложение пересчитывает доли акций в портфеле и обновляет визуализации, показывая текущие рекомендации и их обоснования.

Для получения рекомендаций используется встроенный алгоритм, обученный в параграфе 2.4, который анализирует исторические данные,

текущее состояние портфеля и остаток свободных денежных средств и выбирает какие акции добавить, убрать или оставить.

```python
if st.sidebar.button("Получить рекомендацию"):
    portfolio_test = equal_weights(st.session_state['df_final'])
    agent_results, predictions_value = utils.evaluate_agent(st.session_state['df_predicted'], portfolio_test, cash)
    category_dict = define_recommendation_category(agent_results)
    recommended_category = {
        "К покупке": [key for key, value in category_dict.items() if value == 1],
        "К продаже": [key for key, value in category_dict.items() if value == 2],
        "Держать": [key for key, value in category_dict.items() if value == 0]
    }
    recommended_stocks = {}
    # Выводим карточки в каждой категории
    for category, stocks in recommended_category.items():
        st.subheader(category)
        for stock in stocks:
            price_data = stocks_data[stock]['prices']
            predicted_data = stocks_data[stock]['predicted']
            name = stocks_data[stock]['name']
            recommended_stocks[stock] = category
```

Рис.32. Код распределения рекомендаций.

Сам агент загружается из сохраненного файла, а среда, с которой он взаимодействует реализована в utils.py.

```python
def evaluate_agent(df, portfolio, cash=1000000):
    env = DummyVecEnv([lambda: PortfolioEnv(df, df, portfolio, cash)])
    agent = TD3.load('trained_TD3-5')
    obs = env.reset()
    done = False
    portfolio_values = []
    results = []
    while not done:
        result = {}
        action, _states = agent.predict(obs)
        obs, reward, done, state = env.step(action)
        if done:
            break
        real_state = state[0]['real_state']
        cash, portfolio, data = real_state[:1], real_state[1:41], real_state[41:]
        portfolio_value = np.sum(portfolio * data) + cash
        portfolio_values.append(portfolio_value)
        try:
            actions_dict = dict(zip(df.columns, state[0]['actions']))
        except:
            actions_dict = dict(zip(df.columns, [0] * 40))
        result['cash'], result['portfolio'], result['actions'] = cash, portfolio, actions_dict
        results.append(result)
    return results, portfolio_values
```

Рис.33. Код формирования рекомендаций от агента.

Функция define_recomendation_category определяет какое из трех категорий действий ("Покупать", "Продавать" и "Держать") относится к какой акции. На основе этих рекомендаций формируется список акций для каждой категории, который затем выводится на экран. У карточек акций также выводится рекомендация, полученная от прогнозного алгоритма.

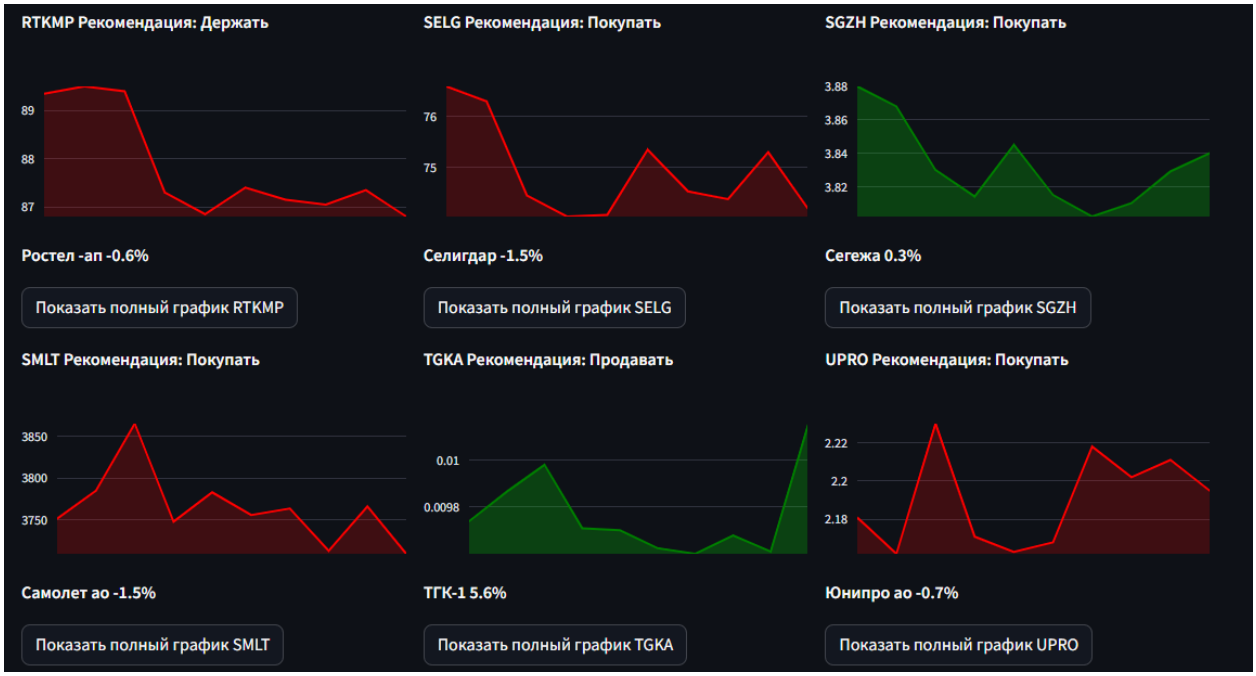

Рис.34. Отображение рекомендации у карточки акции.

Для удобства пользователя в приложении реализованы интерактивные элементы, такие как кнопки и боковые панели, что позволяет легко изменять параметры и получать обновленные данные в режиме реального времени.

Таким образом, внедрение прогнозного алгоритма в приложение на базе Streamlit позволяет создать интерактивный инструмент для управления инвестициями, который сочетает в себе аналитические и визуализационные возможности, обеспечивая удобный и эффективный процесс принятия решений для пользователей. Использование SQLAlchemy и SQLite для хранения данных добавляет надежность и гибкость в управлении пользовательскими данными и конфигурацией системы.

Заключение

Методы машинного обучения и нейронные сети являются эффективными инструментами анализа и обработки больших наборов данных. С их помощью можно повысить эффективность формирования портфеля акций.

Выпускная квалификационная работа имела цель разработать рекомендательную систему для управления инвестиционным портфелем, которая бы являлась эффективным инструментом для автоматизированного формирования и управления инвестиционными портфелями, способного адаптироваться к изменениям на финансовых рынках и предоставлять пользователям рекомендации.

В ходе достижения цели были выполнены следующие задачи:

1. Были изучены современные методов и подходов к прогнозированию цен акций с помощью нейронных сетей.
2. Была разработана рекомендательная система, основанная на предсказаниях цен акций с использованием рекуррентных нейронных сетей (RNN) и методах обучения с подкреплением (RL).
3. Проведена оценка эффективности разработанной системы в условиях реальных рыночных данных и сравнение её с эталонным индексом.

Во время выполнения работы были продемонстрированы возможности использования современных технологий машинного обучения и искусственного интеллекта для решения задач управления инвестициями. Разработанная система не только позволяет прогнозировать цены акций с высокой точностью, но и эффективно использовать эти прогнозы для формирования инвестиционных стратегий, что подтверждается результатами тестирования.

Таким образом, все поставленные в начале работы задачи успешно выполнены, а результаты доказывают, что цель работы была достигнута.

Список использованных источников и литературы

1. Аскинадзи В.М., Максимова В.Ф. Портфельные инвестиции / Московская финансово-промышленная академия. - М., - 2005. – с. 62
2. Elliott, Ralph Nelson, Donald C. Douglas, Matilda W. Sherwood, D. K. Laidlaw, and P. W. K. Sweet. The wave principle.
3. Zhang, G., Eddy Patuwo, B., Hu, M. Y., 1998. Forecasting with artificial neural networks: The state of the art. Int. J.Forecast.
4. H. Hewamalage, C. Bergmeir, K. Bandara, Recurrent neural networks for time series forecasting: current status and future directions, Int. J. Forecast. (ISSN: 0169-2070).
5. C. Olah. Understanding LSTM Networks. сайт - URL: http://colah.github.io/posts/2015-08-Understanding-LSTMs/ (дата обращения 20.01.2024). – Текст: электронный.
6. K. Bandara, C. Bergmeir, S. Smyl, Forecasting across time series databases using recurrent neural networks on groups of similar series: a clustering approach, Expert Syst. Appl. (ISSN: 0957-4174).
7. S. Smyl, A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting, Int. J.Forecast
8. Omer Berat Sezer, Mehmet Ugur Gudelek, Ahmet Murat Ozbayoglu, Financial time series forecasting with deep learning: A systematic literature review: 2005–2019, Applied Soft Computing, Volume 90, 2020, 106181
9. Абдукаева.А.А, Ельшин.Л.А, Гильманов.А.М, Бандеров.В.В./ Прогностические модели развития рынка криптовалюты (ARMA, LSTM): сравнительный анализ // Казанский экономический вестник. – 2019. – №6(44). – С. 88-96.
10. Горшенин, А. К. Анализ конфигураций LSTM-сетей для построения среднесрочных векторных прогнозов / А.К. Горшенин, В. Ю. Кузьмин // Информатика и ее применения. – 2020. – Т. 14. – № 1. – С. 10-16. – DOI 10.14357/19922264200102.

11. Обрубов, М. О. Применение LSTM-сети в решении задачи прогнозирования многомерных временных рядов /М. О. Обрубов, С. Ю. Кириллова // Национальная Ассоциация Ученых. – 2021. – № 68-2. – С. 43-48.
12. Алжеев, А. В. Сравнительный анализ прогнозных моделей ARIMA и LSTM на примере акций российских компаний / А. В. Алжеев, Р. А. Кочкаров // Финансы: теория и практика. – 2020. – Т. 24. – № 1. – С. 14-23. – DOI 10.26794/2587-5671-2020-24-1-14-23.
13. Xinyi Li, Yinchuan Li, Yuanchen Zhan, Xiao-Yang Liu, "Optimistic Bull or Pessimistic Bear: Adaptive Deep Reinforcement Learning for Stock Portfolio Allocation", 2019 – Текст: электронный.
14. Santos, G.C., Garruti, D., Barboza, F., de Souza, K.G., Domingos, J.C. and Veiga, A., Management of investment portfolios employing reinforcement learning, 2023 – Текст: электронный.
15. Liu, X. Y., Xiong, Z., Zhong, S., Yang, H., & Walid, A. Practical deep reinforcement learning approach for stock trading, 2018 – Текст: электронный.