

**Министерство образования и науки Российской Федерации
Московский физико-технический институт (государственный
университет)**

**Физтех-школа прикладной математики и информатики
Кафедра дискретной математики**

Выпускная квалификационная работа магистра

**Применение моделей глубокого обучения для
прогнозирования финансовых временных рядов**

Автор:

Студент М05-217р группы
Зелезецкий Даниил Владимирович

Научный руководитель:

к.ф.-м.н.
Александр Владимирович Куликов

Москва 2024

Оглавление

1 Введение	5
1.1 Актуальность работы	5
1.2 План работы	7
1.3 Цели работы и её значимость	8
2 Теоретическая часть	11
2.1 Понятие, свойства и характеристики финансовых временных рядов	11
2.2 Специфика подбора функции потерь	13
2.3 Предобработка временного ряда	15
2.4 Кросс-валидация	16
2.5 Выбор признакового описания	17
2.6 Обзор моделей прогнозирования	18
2.7 Оптимизация портфеля	22
3 Практическая часть	27
3.1 Данные и их обработка	27
3.2 Исследуемые модели	28
3.3 Функция потерь	31
3.4 Поиск оптимального портфеля	33
3.5 Методология эксперимента	37
3.5.1 Поиск оптимальной архитектуры и набор данных	38
3.5.2 Поиск оптимальных гиперпараметров торговой стратегии	39
3.5.3 Тестирование полученной стратегии	40
4 Результаты работы	43
4.1 Результаты поиска оптимальной архитектуры	43
4.1.1 Итерация 1	49
4.1.2 Итерация 2	50
4.2 Результаты поиска оптимальных гиперпараметров	51
4.2.1 Итерация 1	51
4.2.2 Итерация 2	52
4.3 Результаты тестирования торговой стратегии	53
4.3.1 Итерация 1	54
4.3.2 Итерация 2	56
4.4 Обсуждение результатов	59
4.4.1 Обсуждение результатов поиска оптимальной архитектуры	60
4.4.2 Обсуждение результатов поиска оптимальных гиперпараметров торговой стратегии	60
4.4.3 Обсуждение результатов тестирования стратегии	61
4.4.4 Результаты работы и перспективы её развития	61

Глава 1

Введение

1.1 Актуальность работы

Анализ временных рядов представляет из себя развитую теорию, берущую своё начало с появления первых подходов к анализу временных данных и заканчивая современными методами, включая глубокое обучение. В теории временных рядов существует множество задач и приложений, из которых наиболее распространёнными являются:

1. Прогнозирование финансовых показателей. Одной из основных задач является прогнозирование цен акций, валютных курсов, объемов торгов и других финансовых показателей на основе исторических данных временных рядов [1].
2. Моделирование и прогнозирование экономических показателей. Прогноз экономических показателей, таких как ВВП, инфляция, безработица и объемы производства [2].
3. Прогнозирование погоды и климата. Анализ погоды и климата позволяет делать прогноз погоды на основе исторических данных о температуре, осадках, ветре и других метеорологических параметрах [3].
4. Медицинский мониторинг и прогнозирование заболеваний. Анализ временных рядов в медицине позволяет контролировать здоровье пациентов, прогнозировать развитие заболеваний, выявлять тенденции и паттерны заболеваемости [4].
5. Прогнозирование трафика и спроса. Анализ данных о движении транспорта, спросе на товары и услуги позволяет оптимизировать логистику, планировать ресурсы и принимать решения о предоставлении услуг [5].

Среди всех возможных задач из данной области, особое внимание привлекают задачи прогнозирования цен финансовых активов по причине некоторых особенностей, среди которых можно выделить две основные:

1. Временной ряд цен актива, как правило, не имеет чётких и явно выраженных характеристик, таких как тренд, сезонная и циклическая компоненты, а также постоянная дисперсия и среднее значение. Отсутствие подобных черт делает ряд более непредсказуемым и сложно прогнозируемым [6]. Более того, принято считать, что поведение цены актива схоже с поведением мартингала, стохастического процесса, у которого ожидаемое будущее значение равно его текущему значению, при условии известной информации до настоящего момента. Более подробное рассмотрение этих особенностей будет проведено далее.

-
2. Если рассматривать цену актива с точки зрения финансов как науки, то на её значение напрямую влияют объёмы спроса и предложения, а также механизмы работы биржи. Если принципы работы механизмов доподлинно известны, то факторы принятия решений других игроков о том, покупать или продавать актив скрыты от глаз исследователя, а это значит, что не существует явно выраженных и общепринятых признаков, которые можно использовать при описании ряда в задаче его прогнозирования. Данная проблема также будет рассмотрена в работе далее.

Так или иначе, приведённые выше особенности не подавляют интерес исследователей по всему миру к данной задаче. Наибольший прирост в точности прогноза появился с началом развития моделей глубокого обучения, поскольку именно они продемонстрировали, что способны находить сложные паттерны во входящих последовательностях, которые были недоступны лаконичным, но от этого и ограниченным базовым моделям. Различные методы глубокого обучения, такие как алгоритмы полносвязных, рекуррентных и сверточных нейронных сетей стали широко применяться для анализа и прогнозирования временных данных в различных областях, включая финансы, экономику, медицину, климатологию и другие. Следует также отметить, что популярность нейросетевых архитектур начала набирать обороты по мере развития вычислительных мощностей компьютеров, а именно их процессоров и графических ускорителей. Современные методы анализа временных рядов позволяют обрабатывать большие объемы данных, выявлять скрытые закономерности и паттерны, предсказывать будущие тенденции и принимать обоснованные решения на их основе.

Глубокое обучение позволило получить прирост точности благодаря своей возможности извлекать сложные зависимости из данных. Начиная с простых рекуррентных и сверточных нейронных сетей, исследователи постоянно совершенствуют модели глубокого обучения для более точного прогнозирования. С появлением LSTM (Long Short-Term Memory) [7] и GRU [8] (Gated Recurrent Unit) сетей глубокое обучение стало способным эффективно учитывать долгосрочные зависимости во временных рядах.

Благодаря схожести постановок задач прогнозирования временных рядов и анализа языка, практически все архитектуры из одной области могут быть применены в другой с незначительными доработками [9]. Машинная обработка естественного языка является активно развивающейся областью, которая получила значительный прирост внимания, благодаря статье [10] и предложенной в ней архитектурой Transformer с блоками внимания, именуемыми авторами как Multi-Head Attention Layer и способными особенно качественно захватывать паттерны в данных, о чём свидетельствуют приложенные в статье результаты. Данная архитектура также может быть применена в задаче прогнозирования финансовых рядов. Обоснованием актуальности применения Трансформеров в задаче прогнозирования финансовых данных служат Multi-Head Attention блоки, которые будут исследоваться на способность качественного обобщения входящих последовательностей цен. Другие исследования частично подтверждают гипотезу об эффективности трансформеров на финансовых данных. Например, в работе [11], автор применяет комбинацию свёрточной архитектуры и трансформера в задаче прогнозирования направления движения рынка. Результаты статьи свидетельствуют об эффективности данного подхода.

Более того, актуальность применения глубокого обучения в прогнозировании цен акций обосновывается повышенными возможностями нейросетевых моделей по сравнению с классическими подходами машинного обучения [12], а значит, потенциально более высокими доходностями в торговле.

1.2 План работы

Данную работу можно разделить на следующие этапы:

Этап 1: Первым этапом данной работы является исследование и сравнение современных моделей глубокого обучения при решении задачи прогнозирования цен акций Российских компаний. В центре внимания данной работы находятся такие архитектуры, как Transformer [10] и TSMixer предложенная в работе [13]. В качестве уже зарекомендовавшего себя подхода используется архитектура LSTM [14], с эффективностью которой будут сравниваться модели, упомянутые выше. В качестве данных используются почасовые цены акций компаний, входящих в индекс Московской Биржи на момент проведения исследования. В качестве признакового описания ряда будут использоваться как технические индикаторы, так и значения цен других временных рядов. В качестве первых, будут использоваться 57 технических индикаторов, рассчитанных на основе всей доступной информации по стакану заявок и полученных с помощью официальной библиотеки MoexAlgo, написанной на языке программирования Python [15]. В качестве цен других активов будут использоваться цены закрытий 42 компаний, входящих в Индекс Мосбиржи. В ходе исследования, описанные выше модели пройдут серию из трех испытаний, приведённых в таблице ниже:

Таблица 1.1: Таблица испытаний

	И.1	И.2	И.3
LSTM	•	•	•
Transformer(encoder часть)	•	•	•
Transformer(encoder + decoder)	•	•	•
TSMixer	•	•	•

где:

И.1: Одномерный ряд(без признаков), прогноз на 1 час вперёд.

И.2: Многомерный ряд(признаки - технические индикаторы), прогноз на 1 час вперёд.

И.3: Многомерный ряд(признаки - другие активы на рынке), прогноз на 1 час вперёд.

В рамках данного этапа, модели проходят серию испытаний с дальнейшим сравнением по значению функции потерь. Таким образом, этап позволяет нам найти оптимальную с точки зрения точности прогнозов комбинацию 'модель-признаковое описание'. Данная комбинация переходит во второй этап оптимизации торговой стратегии.

Этап 2: Опираясь на полученные результаты, второй этап представляет из себя построение и оптимизацию торговой стратегии по её гиперпараметрам. Метрикой качества стратегии будет являться коэффициент Шарпа. Комбинация гиперпараметров, дающая наибольшее значение коэффициента Шарпа, будет применена в стратегии и протестирована на следующем этапе.

В ходе работы алгоритма, каждый час производится перебалансировка портфеля путём решения задачи оптимизации. Более подробная постановка задачи и детальное описание алгоритма приводится в главе 2.

Этап 3: После нахождения оптимальных гиперпараметров торгового алгоритма, будет произведено его тестирование на тестовых данных, на которых не оптимизировались ни модель ни сам алгоритм. Проверка будет проводиться путём бектестинга стратегии.

Общая схема всех трёх этапов представлена на рисунке 3.7.

1.3 Цели работы и её значимость

Целью данной работы является проверка эффективности ряда нейросетевых архитектур в задаче прогнозирования цен акций компаний Индекса Мосбиржи и оценка их качества в условиях торговли.

Задачами исследования являются: формирование данных для исследования, разработка функции потерь для обучения модели, проведение перечня экспериментов и отбор наиболее успешной конфигурации, поиск оптимальных гиперпараметров торговой стратегии, проведение тестирования оптимальной стратегии путём бектестинга.

Важно отметить, что исследуемые архитектуры Transformer и TSMixer являются state-of-the-art подходами и на большом количестве бенчмарков установили рекорды точности либо вошли в список лидеров. Модель Transformer изначально разрабатывалась для решения задач языкового моделирования и продемонстрировала высокое качество благодаря блокам внимания Multi-Head attention layer, которые и по сей день являются фундаментальной основой таких популярных языковых архитектур, как GPT [16]. Лишь немногим позднее модель получила свою адаптацию для работы в временными рядами. В то же время, TSMixer изначально разрабатывался для работы с многомерными временными рядами.

Говоря о первой модели, можно выделить следующие её достижения:

1. WMT 2014 English-to-German Translation: Трансформеры показали выдающиеся результаты на этом бенчмарке, превзойдя предыдущие модели и демонстрируя высокий уровень качества машинного перевода [10].
2. GLUE Benchmark: На этом наборе задач для оценки качества моделей в области обработки естественного языка, трансформеры показали отличные результаты, превзойдя другие архитектуры [17].
3. SQuAD Benchmark: В задаче вопрос-ответ на базе текста, трансформеры также демонстрировали высокую точность и эффективность, ставя рекорды на данном бенчмарке [18].
4. ImageNet: Хотя трансформеры изначально разрабатывались для обработки текста, они также показали потенциал в области обработки изображений, превзойдя некоторые традиционные архитектуры на бенчмарке ImageNet [19].
5. M4 Competition: Трансформеры были успешно применены к прогнозированию рядов в рамках крупнейшего соревнования M4 Competition и показали конкурентоспособные результаты по сравнению с другими архитектурами [20].

Архитектура TSMixer обладает неоспоримым преимуществом в виде простоты своей архитектуры, которая будет обсуждаться далее. В популярных академических тестах, TSMixer сравним со специализированными современными моделями, которые

используют индуктивные искажения конкретных тестов. В сложном и крупномасштабном тесте M5 на наборе данных для розничной торговли, TSMixer демонстрирует большую производительность по сравнению со state-of-the-art альтернативами. В работе [13] приведены результаты точности данной модели.

Все перечисленные выше достижения побуждают интерес к применению этих моделей для анализа финансовых данных, так как можно предположить, что предложенные архитектуры смогут вылавливать сложные закономерности, которые расположены далеко друг от друга во входящей последовательности и не имеют шаблонную структуру, описанную ранее. Всё это подтверждает актуальность настоящей работы, а также её новизну и значимость. Более того, как утверждалось в описании первого этапа работы, в качестве признаков временных рядов цен будут использоваться 57 технических индикаторов, библиотеку для получения которых представила Московская Биржа в октябре 2023 года [15].

В ходе работы, автором использовались исключительно официальные источники тренировочных и тестовых данных. В качестве источников информации использовались научные статьи, материалы из учебников, которые использовались в образовательных целях либо официальные сайты организаций. Все расчёты производились на языке программирования Python. На каждом этапе обработки данных, автором проводилась проверка на корректность их обработки. Полученные в ходе исследований результаты подтверждаются другими работами в данной области и не противоречат никаким теоретическим и практическим результатам, которые будут обсуждаться в следующей части.

Глава 2

Теоретическая часть

2.1 Понятие, свойства и характеристики финансовых временных рядов

Временной ряд представляет собой упорядоченный набор данных, измеренных в последовательные моменты времени. Эти данные могут быть собраны с определенной периодичностью, например, ежедневно, еженедельно, ежемесячно или в ином временном разрезе. Основное свойство временных рядов заключается в их зависимости от времени, что отличает их от обычных независимых и однородных наборов данных.

Временные ряды широко используются в различных областях для анализа и прогнозирования. Например, в экономике они могут отражать динамику финансовых рынков, показатели инфляции, объемы производства и другие важные экономические показатели. В метеорологии временные ряды могут содержать информацию о погоде, климатических изменениях и т.д. Для анализа временных рядов применяются различные методы, включая статистические методы, машинное обучение, и эконометрику. Целью анализа является выявление закономерностей, трендов, сезонных колебаний, цикличности и других характеристик данных, а также построение моделей для прогнозирования будущих значений временного ряда.

Временные ряды обладают рядом свойств, наиболее важные из которых:

1. Зависимость от времени. Основное свойство временных рядов заключается в том, что значения в ряду упорядочены по времени.
2. Тренд. Тренд представляет собой общее направление изменения временного ряда в течение длительного некоторого времени. Он может быть восходящим, нисходящим или отсутствовать вообще.
3. Сезонность. Сезонность отражает циклические изменения во временном ряде, которые повторяются в течение определенного периода времени. Например, сезонные колебания могут наблюдаться ежегодно, ежемесячно.
4. Цикличность. Цикличность представляет собой изменения временного ряда, которые происходят не с фиксированной периодичностью, но имеют длительные циклы изменений.
5. Шум (случайная составляющая). Шум или случайная составляющая временного ряда представляет собой непредсказуемые и случайные колебания, которые могут быть вызваны различными факторами и вносят некоторую степень неопределенности в данные.

-
- 6. Стационарность. Стационарность временного ряда означает, что его статистические свойства (среднее, дисперсия, автокорреляция и т.д.) не меняются со временем. Это важное свойство, которое облегчает анализ и прогнозирование временных рядов для некоторых моделей.
 - 7. Автокорреляция. Автокорреляция представляет собой степень линейной зависимости между значениями временного ряда в различные моменты времени. Высокий уровень автокорреляции может указывать на наличие закономерностей в данных и использоваться для прогнозирования.
 - 8. Среднее. Среднее значение временного ряда отражает общую тенденцию данных и может быть использовано для определения тренда или базового уровня ряда.
 - 9. Дисперсия. Дисперсия временного ряда измеряет степень изменчивости данных вокруг их среднего значения. Высокая дисперсия может указывать на большую изменчивость данных, в то время как низкая дисперсия указывает на более стабильные значения ряда.

Временной ряд цен финансовых активов далеко не всегда имеет некоторые перечисленные свойства. Например, типичный ряд цены акции не имеет постоянного тренда, так как ему свойственно регулярно меняться, в нём не наблюдается явной сезонной составляющей и циклической компоненты, так как доподлинно неизвестны истинные факторы, влияющие на настроения игроков. Более подробное обсуждение свойств финансовых временных рядов приводится в работах [21] [22]

Так как в данной работе в качестве данных используются цены акций входящих в состав Индекса Мосбиржи, то следует остановить внимание на общей проблематике их прогнозирования.

Как упоминалось ранее, временные ряды цен акций не имеют выраженного тренда, сезонности, постоянного среднего и дисперсии, тем самым усложняя процесс прогнозирования. Более того, нет общезвестных факторов, которые влияют на решения экономических агентов о продаже/покупке того или иного актива. Ниже приводятся некоторые рассуждения об основных факторах, влияющих на решение игрока:

- 1. Рыночные настроения. Игроки могут реагировать на общие настроения на рынке, которые могут быть вызваны экономическими отчетами, политическими событиями или даже изменениями в социальном настроении.
- 2. Технические индикаторы. Независимо от долгосрочных или краткосрочных стратегий, многие игроки используют технические индикаторы, которые помогают идентифицировать потенциальные точки входа и выхода на рынке.
- 3. Фундаментальный анализ. Фундаментальные факторы, такие как прибыль компании, новости о корпоративных слияниях или поглощениях, изменения в экономической политике или даже изменения в процентных ставках, могут служить стимулом для принятия решений.
- 4. Реакция на новости. События, такие как политические изменения, экономические отчеты или даже непредвиденные события (например, природные катастрофы), могут влиять на рыночные цены.
- 5. Психологические факторы. Страх, жадность и другие эмоциональные составляющие также влияют на решения игроков.

-
6. Стратегии управления риском. Различные методы управления риском, такие как установление стоп-лоссов или диверсификация портфеля, могут стимулировать решения о покупке или продаже как часть общей стратегии управления рисками.

Все приведённые выше рассуждения можно свести к выводу о том, что не существует единых факторов, которые бы достаточно качественно описывали поведение всех игроков сразу, в противном случае весь рынок был бы сильно предсказуем. В рамках данной работы, ввиду высокой частоты обновления портфеля, было принято решение об использовании технических индикаторов в качестве дополнительной информации при прогнозировании цен. Кроме того, эффективность использования технических индикаторов в высокочастотной торговле подтверждается эмпирическими исследованиями [23]. Так, авторы работ [24] и [25] демонстрируют эффективность работы нейронных сетей с техническими индикаторами для предсказания цен акций. В работе [26], авторы предлагают торговую стратегию, основанную на использовании нейросетевых подходов с использованием технических индикаторов как признаков. Предложенная система показывает годовую доходность в 15.99% против стратегии Buy and Hold, которая принесла за это же время 11.05% годовых.

2.2 Специфика подбора функции потерь

Следующей особенностью нейросетевого прогнозирования цен акций является выбор корректной функции потерь при обучении модели. В большинстве задач прогнозирования, роль функции потерь выполняет Mean Squared Error.

$$MSE(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2 \quad (2.1)$$

Действительно, данная функция квадратично штрафует модель за неверный прогноз вне зависимости от знака ошибки, тем самым заставляя её приближаться к истинному значению. Однако, как только в задаче прогнозирования появляется спекулятивный интерес, то MSE функция сама по себе перестаёт отвечать всеменным требованиям. Прогнозирование цен акций является не самоцелью, а лишь инструментом получения достоверных сигналов к покупке/продаже актива, а следовательно, крайне важным становится не только приблизить прогноз к истинному значению, но и угадать направление движения актива в прогнозируемом периоде(угадать знак доходности). Подробнее на примере:

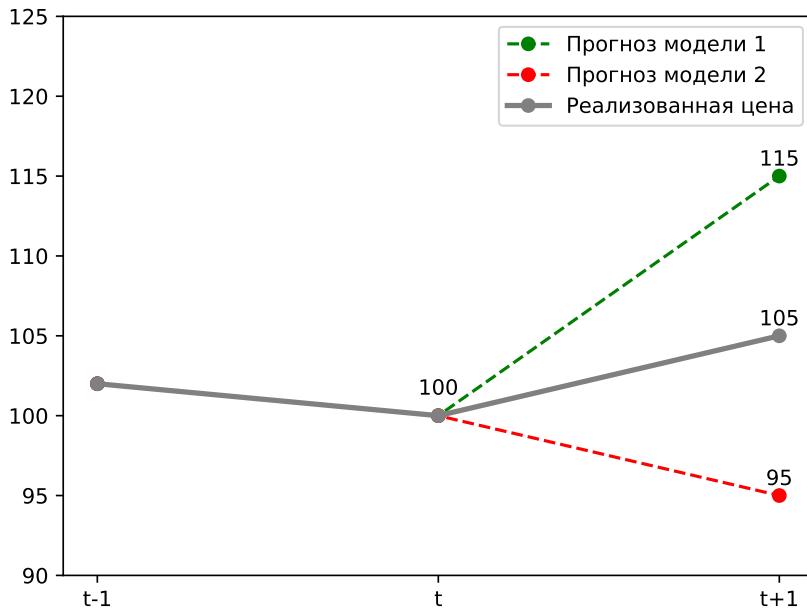


Рис. 2.1: Прогнозные модели и реализованная цена

$$Loss(y_{i+1}, \hat{y}_{i+1}) = (y_{i+1} - \hat{y}_{i+1})^2 - \lambda(\hat{y}_{i+1} - y_i)(y_{i+1} - y_i)$$

$$Winrate(y, \hat{y}) = \frac{1}{N} \sum_{i=2}^N [sign(y_i - y_{i-1}) = sign(\hat{y}_i - y_{i-1})] \quad (2.2)$$

Как видно на рисунке 2.1 в момент времени t цена актива достигает 100. В этот же момент две модели делают свой прогноз цены на $t+1$ период. Модель 1 прогнозирует рост до 115, модель 2 прогнозирует падение до 95, в то время как реальная цена оказалась равна 105. Подставляя ошибки обеих моделей в MSE функцию, мы получим значение MSE=100 в обоих случаях, тем самым уровняя их ошибку по своей значимости. В действительности же, цена ошибки может существенно отличаться: используя прогноз модели 2 как сигнал для открытия короткой позиции с последующим выкупом акции в момент $t+1$ мы потеряем деньги. В свою очередь, расценив прогноз модели 1 как сигнал к покупке, мы получим более низкую доходность к $t+1$ периоду времени, так как прогноз оказался выше истинного значения, однако в силу того, что само направление движения угадано, мы не теряем средств во всех случаях кроме тех, когда реализованная доходность оказалась меньше, чем понесённые транзакционные издержки.

Эти рассуждения наталкивают на вывод о том, что при спекулятивном интересе на первый план выходит не только точность модели, но и её способность угадывать направление движения ряда. Среднеквадратическая функция потерь не удовлетворяет накладываемым требованиям, а значит должна быть либо модернизирована либо заменена вовсе. В работе [27], автор предлагает ряд функций потерь для прогнозирования доходностей, которые штрафуют модель за некорректно угаданный

знак. Вот несколько из них:

$$Loss(y, \hat{y}) = -sign(\hat{y}) \cdot y \quad (2.3)$$

$$Loss(y, \hat{y}) = -sign(\hat{y}) \cdot sign(y) \quad (2.4)$$

где

$$sign(x) = \begin{cases} -1, & \text{если } x \leq 0, \\ 0, & \text{если } x > 0. \end{cases}$$

Как можно заметить, обе функции работают с прогнозами доходностей, а не цен. Опираясь на их основную идею, в рамках данной работы была придумана адаптация к ценовому формату прогнозов. Более подробное описание разработанной функции потерь приводится в главе 3.

2.3 Предобработка временного ряда

Методология обработки временных рядов, как финансовых так и остальных, для большинства методов прогнозирования неизменна. Фокусируясь на задаче обработки рядов для нейросетевого прогнозирования, следует обратить внимание на следующие подходы:

Удаление выбросов: Данное решение направлено на очистку данных от выбросов, которые могли произойти как по причине некорректного сбора данных, так и по причине их особенностей. Под выбросом будем считать значение, лежащее на расстоянии более чем два стандартных отклонения от среднего того распределения, из которого оно пришло.

Нормализация или стандартизация данных: Разные шкалы признаков могут затруднить обучение нейросетей, поскольку модели часто оказываются чувствительны к масштабу входных данных. Более того, при больших значениях ряда градиент функции потерь зачастую затухает, так как производная по функции активации близка к нулю. Наиболее популярными методами такой предобработки являются нормализация

$$MinMaxScaler(X) = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (2.5)$$

и стандартизация

$$StandardScaler(X) = \frac{X - \mu}{\sigma^2} \quad (2.6)$$

Среди нейросетевых методов обработки ряда стоит отметить работу [28], авторы которой предлагают использовать специально разработанный EDAIN слой, который адаптируется к подающимся в него данным и вырабатывает индивидуальные параметры нормализации ряда, тем самым улучшая точность прогноза.

Сглаживание: Техники сглаживания, такие как скользящее среднее или экспоненциальное сглаживание, помогают уменьшить волатильность временных рядов и выделить более значимые тренды и циклы. Данные техники часто используются на сильно зашумлённых временных рядах.

Декомпозиция: Многие временные ряды содержат тренды и сезонные компоненты, которые могут влиять на прогнозы. Их идентификация и корректировка или

декомпозиция на составляющие позволяют модели лучше уловить основные закономерности в данных. В работе [29], авторы предлагают нейросетевой метод декомпозиции ряда, который, согласно приведённым результатам, позволяет достичь более точного прогноза на некоторых задачах.

Обработка пропусков в данных: Заполнение пропущенных значений важно для обеспечения непрерывности и полноты временных рядов, что критично для прогнозирования. По мере развития глубокого обучения, стали набирать популярность обучаемые методы заполнения пропусков. Так, нейросетевой алгоритм SAITS, опубликованный в работе [30] основан на механизме self-attention, который будет обсуждаться далее. Обширные эксперименты количественно и качественно демонстрируют, что SAITS превосходит классические методы в задаче заполнения пропусков во временных рядах.

Обработка временного ряда является неотъемлемой частью общего процесса, позволяющей добиться более качественных результатов. Более подробное описание процесса предобработки изложено в главе 2.

2.4 Кросс-валидация

Кросс-валидация представляет из себя метод оценки эффективности модели, позволяющий проводить её тестирование на разных частях данных. В отличие от обычного разделения данных на тренировочную и тестовую выборки, кросс-валидация даёт более надёжную оценку качества модели. Наиболее распространённым видом кросс-валидации является метод K-Fold, который представляет из себя деление данных на K одинаковых частей, при этом обучение происходит на K-1 части и тестирование на оставшейся. Данный процесс происходит итеративно, пока не протестируются все K участков с обучением на K-1 соответственно. В конце результаты тестирования усредняются на константу K. Данный подход позволяет оценить качество модели в совокупности на всех частях данных.

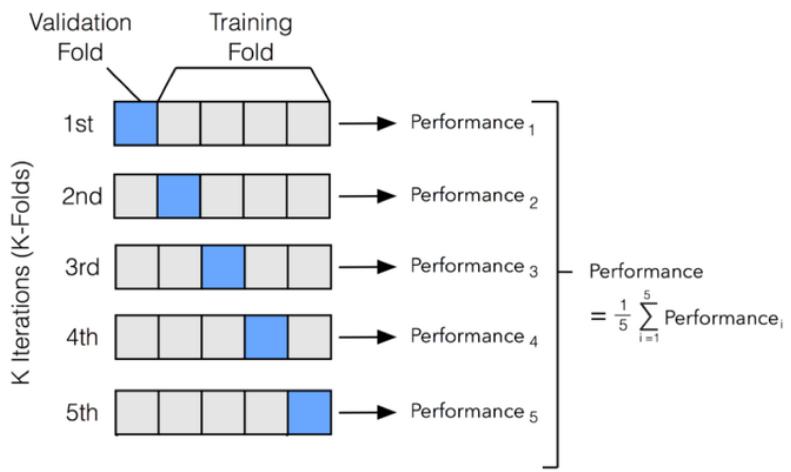


Рис. 2.2: K-Fold кросс-валидация

Однако, данный подход неприменим при работе с временным рядом, так как допускает утечку данных из тренировочного набора в тестовый, а значит, модель начинает получать дополнительную информацию, что пагубно влияет на достоверность оценки. В работе [31], автор предлагает модифицированный метод кросс-валидации Purged Cross-Validation, учитывающий отступы между тренировочными и тестовыми

наборами данных. Размер отступов равен длине исторического среза данных, на основании которого модель делает прогноз. Данный метод позволяет избежать утечки информации и сделать процесс проверки модели более корректным с методологической точки зрения.

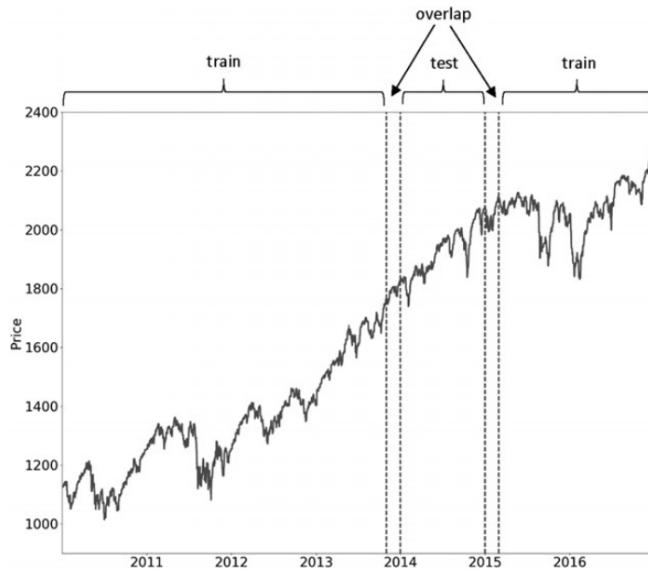


Рис. 2.3: Purged кросс-валидация

2.5 Выбор признакового описания

Признаковое описание временных рядов важно для эффективного анализа ряда и позволяет значительно улучшить качество прогноза. Качественный набор признаков при работе с финансовыми данными зависит от их доступности и частоты обновления ряда. Так, например, решая задачу долгосрочного инвестирования на первый план выходят показатели финансовой отчётности и сопутствующие индикаторы, такие как рентабельность активов (ROA), рентабельность собственного капитала (ROE), коэффициент текущей ликвидности, коэффициент соотношения собственного и заемного капитала, показатель Цена/Прибыль (P/E) и прочие. Решая задачу краткосрочного прогнозирования, на первый план выходят признаки в виде лаговых переменных, скользящих статистик, временных меток и индикаторов из внешних источников. Поскольку частота обновления наших данных составляет один час, то разумно предположить, что эффективнее всего себя проявят именно технические индикаторы.

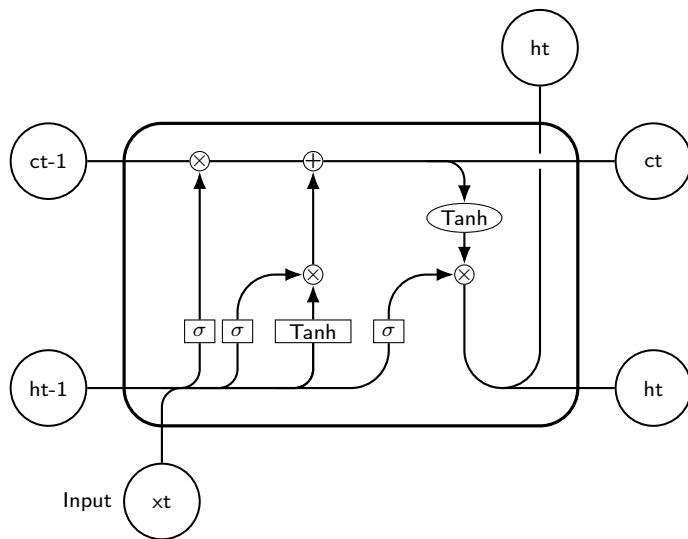
Вопрос предобработки технических индикаторов остаётся дискуссионным, за время проведения исследования общепринятых подходов обнаружено не было. Так, например, автор работы [32] считает, что нормализация, стандартизация или иные методы нормировки технических индикаторов должны строго зависеть от природы самих индикаторов: если индикатор напрямую не связан с ценой и изменяется в узких границах, то он должен быть обработан отдельно от цены, то есть значения цены актива не должны вносить свой вклад в параметры нормировки. И наоборот, технические индикаторы связанные с ценой должны быть преобразованы вместе с самой ценой.

2.6 Обзор моделей прогнозирования

На сегодняшний день, область прогнозирования временных рядов активно развивается, порождая большое количество новых нейросетевых моделей, созданных или адаптированных для их прогноза. Так или иначе, в основе большинства из них лежат уже знакомые архитектуры типа LSTM, Transformer или свёрточных сетей. Обсудим подробнее каждую из них, а далее рассмотрим наиболее эффективные методы использования этих сетей.

LSTM:

Одна из наиболее старых и популярных рекуррентных архитектур, долгое время являвшаяся общепринятым стандартом, использовавшимся при решении задач обработки и генерации числовых последовательностей. Впервые предложенная в 1997 году Зеппом Хохрайтером и Юргеном Шмидхубером в работе [14], она смогла частично решить главную проблему всех предыдущих рекуррентных архитектур - короткую память.



LSTM блок обладает двумя каналами памяти: краткосрочным и долгосрочным. В процессе обучения, данная архитектура подбирает свои веса таким образом, чтобы оптимально сохранять всю возможную информацию последовательности в долгосрочном канале и вытаскивать её в краткосрочный тогда, когда это необходимо.

Данный блок частично решает одну из основных проблем рекуррентных архитектур, однако параллельно сталкивается с проблемой тяжести вычисления при использовании и обучении. Расчёт градиентов сквозь LSTM блок является довольно дорогостоящим процессом с точки зрения вычислительных ресурсов.

Работа [33] посвящена использованию LSTM архитектуры в задаче прогнозирования промышленного индекса Доу-Джонса(DJIA) и трёх наборов курсовых связок доллара США с валютами Индии, Японии и Южной Кореи. Согласно результатам экспериментов, подход авторов позволил им достичь высокой точности на всех четырёх задачах и показать более низкое значение RMSE по сравнению с остальными моделями машинного обучения.

Сравнивая LSTM с более консервативными эконометрическими моделями, следует выделить исследование [34], авторы которого сравнивают её точность с показателями авторегрессионной модели ARIMA [35]. В качестве данных, исследователи используют доходности индексов N225, IXIC, HSI, GSPC, DJI и прочих. Результаты показали, что нейросеть обогнала ARIMA в точности на 85 процентов.

Свёрточная архитектура: Свёрточные нейронные сети (CNN) - это класс ней-

ронных сетей, широко применяемый в обработке изображений и анализе видео. Кроме того, они могут быть адаптированы для задач обработки естественного языка, анализа временных рядов. Их история берёт начало с 1980-ых годов, когда впервые была представлена архитектура LeNet-5 [36] для распознавания рукописных символов.

AlexNet [37], разработанная Алексеем Крижевским и Ильей Светуньковым, была одной из прорывных свёрточных архитектур. Представленная в 2012 году, она выиграла соревнование ImageNet Large Scale Visual Recognition Challenge (ILSVRC) с значительным отрывом от других моделей показав, что нейросетевые методы способны решать задачи классификации изображений с более высокой точностью по сравнению с классическими моделями машинного обучения.

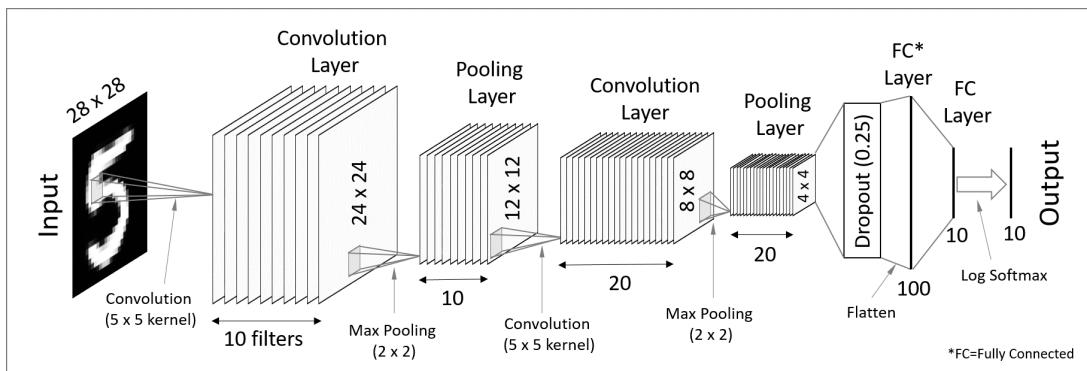


Рис. 2.4: Свёрточная архитектура

Схема типичной свёрточной сети изображена на рисунке 2.4. На вход подаётся трёхмерный тензор, описывающий работу пикселей изображения, а затем ядра свёртки шаг за шагом обрабатывают его, пытаясь обнаружить необходимые для задачи признаки. В определённый момент времени полученный тензор распрямляется в вектор, который подаётся в полно связанный слой.

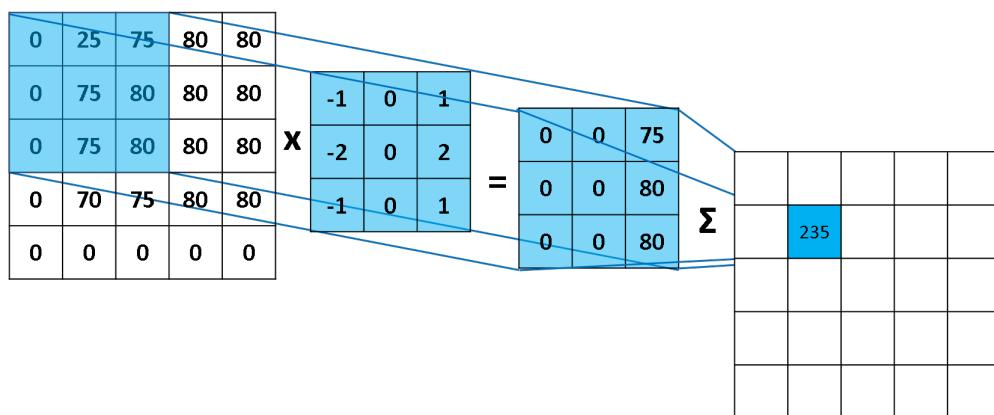


Рис. 2.5: Пример работы ядра свёртки

Как можно видеть на рисунке 2.5, ядро представляет из себя матрицу весов, на которые поэлементно умножается тензор. Веса внутри ядра свёртки являются обучаемыми параметрами, в то время как размер всех ядер задаётся экзогенно и фиксируется на период обучения.

Адаптация свёрточных сетей для прогнозирования ряда может быть разной, трансформация данных зависит от желания исследователя. В самом простом варианте ряд может быть представлен как одномерный или многомерный вектор в зависимости от наличия признакового описания.

Transformer: Наиболее высоких результатов прогнозирования, научное сообщество начало добиваться с появлением архитектуры Transformer, которая вобрала в себя все сильные стороны рекуррентных и свёрточных сетей и одновременно с этим победила их недостатки.

Впервые представленная коллективом исследователей из Google [10], архитектура продемонстрировала наилучшие результаты в задаче машинного перевода, а немногим позднее начала адаптироваться под различные задачи, начиная с обработки фото заканчивая анализом звуковых сигналов.

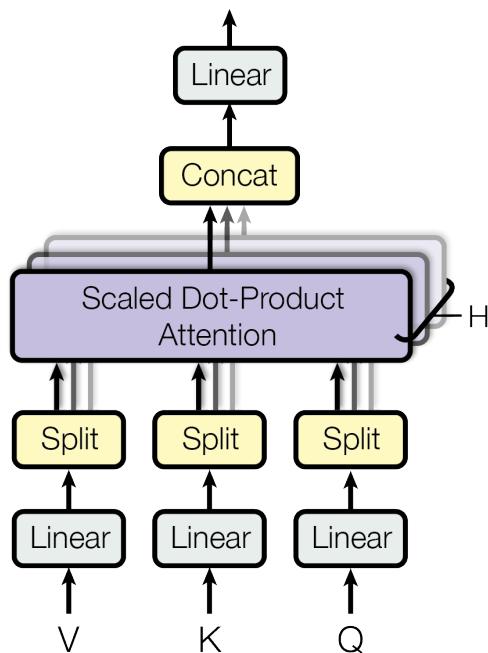


Рис. 2.6: Механизм Attention

Нарисованный на рисунке 1.6 механизм внимания позволяет трансформеру захватывать зависимости значений входящей последовательности при абсолютно любом расстоянии между ними. Аналитическая форма записи данного механизма имеет следующий вид:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (2.7)$$

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O \quad (2.8)$$

$$\text{где } \text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Из-за сходства форматов языковых задач и задач прогнозирования рядов, посредством некоторых адаптаций модель может быть применена в нашей постановке. Более того, у исследователя есть выбор между использованием одного лишь энкодера или полноценной связки энкодер + декодер.

Так, Neo Wu et al. [38] используют канонический формат трансформера с энкодером и декодером. Единственные преобразования находятся в Task-specific слое, где

на выходе стоит один нейрон предсказывающий значение ряда. Решая задачу предсказания эффективности распространения гриппа, автор сравнивает трансформер с моделями ARIMA, LSTM, Seq2Seq+Attention. Согласно результатам, трансформер лидирует по RMSE метрике с большим отрывом.

Статья "Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting-[39]" представляет новую архитектуру "Informer" которая была разработана для решения задач прогнозирования временных рядов с длинными последовательностями данных. Informer также базируется на трансформер архитектуре, однако имеет некоторые модификации. В частности, авторы предлагают механизмы адаптивного управления вниманием и механизм мультишкалируемого внимания, которые позволяют модели обрабатывать длинные временные последовательности более эффективно. Также в работе предлагается механизм адаптивного прогнозирования, который учитывает различные характеристики временных рядов и позволяет модели адаптироваться к различным структурам данных.

Проводя 5 экспериментов на различных рядах(электричество и погода), авторам удалось добиться наилучших результатов по сравнению с моделями LogTrans, Reformer, LSTMa, DeepAR, ARIMA и Prophet четыре из пяти раз.

Авторы модели FEDformer [40] подошли к рассмотрению временного ряда с точки зрения теории обработки сигналов. Основная идея FEDformer заключается в том, чтобы улучшить способность трансформеров к моделированию долгосрочных зависимостей во временных рядах путем внедрения дополнительной информации о частотах. Для этого в модели используется механизм преобразования Фурье, который разделяет входные данные на наборы различных частотных диапазонов. Это позволяет модели эффективно моделировать как короткие, так и долгосрочные зависимости в данных. Обучаясь на данных по электричеству, пробках, погоде и т.д., FEDformer демонстрирует наилучший результат на всех задачах по сравнению с LogTrans, Reformer, Autoformer, Informer.

Решая задачу прогнозирования цен SP500, авторы [11] соединяют свёрточный подход с трансформером воедино, называя это CTTS.

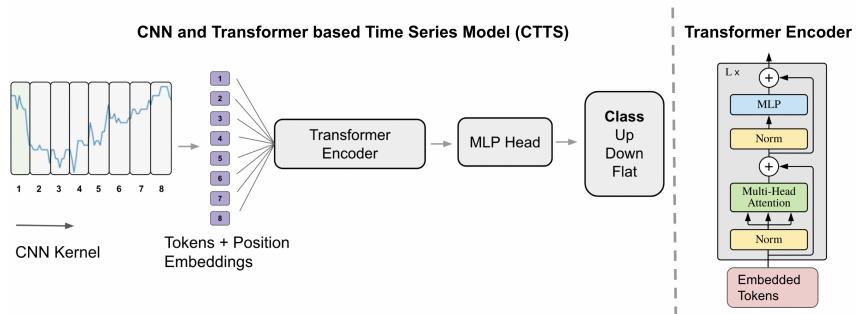


Рис. 2.7: CTTS архитектура

Как можно наблюдать на схеме, сперва свёрточное ядро пробегает по временному ряду(а точнее его векторной репрезентации), затем используя позиционные эмбеддинги модель подаёт данные в энкодер-блок трансформера. Используя выходы энкодера, полно связанный слой прогнозирует поведение цен актива. Согласно результатам, данных подход позволяет угадывать дальнейшее поведение актива с точностью, доходящей в некоторых случаях до 66 процентов.

Как можно увидеть из обзора моделей, наиболее передовой и эффективной архитектурой является Transformer, разные модификации которого показывают наилучшие результаты точности во многих экспериментах. Данный факт обосновывает

актуальность работы, так как особый интерес представляет проверка эффективности этой архитектуры на финансовых временных рядах, особенности которых были описаны ранее.

2.7 Оптимизация портфеля

Процесс прогнозирования цен акций, тонкости которого обсуждались в предыдущих пунктах, является лишь частью торговой системы. Второй шаг - это создание алгоритма оптимизации портфеля, который формирует торговую стратегию. Оптимизация портфеля — это процесс выбора наилучшего распределения активов среди различных финансовых инструментов для максимизации прибыли и минимизации риска.

Теория портфеля Гарри Марковица [41], представленная в 1952 году, заложила основы современной теории портфельных инвестиций и принципов диверсификации. В своей работе, Марковиц опирался на четыре основных принципа: ожидаемую доходность портфеля, его риск, эффективную границу и выбор портфеля с учетом отношения к риску. Опишем их более формально:

Ожидаемая доходность портфеля: Рассчитывается как взвешенная сумма ожидаемых доходностей отдельных активов (R_i), где w_i - доля актива i в портфеле:

$$R_p = \sum_{i=1}^n w_i R_i \quad (2.9)$$

Риск портфеля: Измеряется стандартным отклонением или дисперсией взвешенной суммы доходностей, где ключевым элементом является ковариация между доходностями двух активов:

$$\sigma_p^2 = \sum_{i=1}^n \sum_{j=1}^n w_i w_j \sigma_{ij} \quad (2.10)$$

Эффективная граница: Состоит из портфелей, которые минимизируют риск σ_p для данного уровня ожидаемой доходности R_p или максимизируют R_p для заданного уровня риска. Это можно выразить как решение задачи оптимизации:

$$\min_w (\sigma_p^2) \quad \text{s.t.} \quad \sum_{i=1}^n w_i R_i = R_p \text{ and } \sum_{i=1}^n w_i = 1 \quad (2.11)$$

Выбор портфеля с использованием кривых безразличия: Кривая безразличия описывает уровни риска и доходности, которые считаются эквивалентными для инвестора. Каждая кривая параметризуется уровнем полезности U , который зависит от доходности R_p и риска σ_p :

$$U(R_p, \sigma_p) = R_p - \frac{1}{2} \lambda \sigma_p^2 \quad (2.12)$$

где λ — параметр, выражющий степень чувствительности к риску инвестора.

Индексные модели, появившиеся в 1970-х годах, были разработаны для упрощения расчетов, связанных с оценкой риска и доходности в модели портфеля Марковица. Они помогают уменьшить число необходимых оценок ковариации между активами, что упрощает процесс оптимизации портфеля. Идеей таких моделей является

выражение доходности всех активов в портфеле через доходности общего рыночного индекса. Предполагается, что доходность каждого актива зависит рыночным индексом и специфическими факторами, некоррелирующими с рынком. В простейшем виде индексная модель имеет следующий вид:

$$R_i = \alpha_i + \beta_i R_m + \epsilon_i \quad (2.13)$$

где :

R_i - доходность актива i

α_i - константа, отражающая среднее значение доходности актива i , не зависящее от рынка

β_i - коэффициент, показывающий чувствительность доходности актива i к изменениям рыночной доходности R_m

R_m - доходность рыночного индекса

ϵ_i - случайная ошибка, представляющая специфические факторы актива, не связанные с рыночными движениями.

Преимуществами индексных моделей можно назвать простоту расчётов и возможность фокусировки на рыночных факторах.

Предложенная Шарпом в 60-х годах модель оценки капитальных активов (Capital Asset Pricing Model, CAPM) [42] базируется на предположении о том, что инвесторы требуют дополнительную доходность (премию за риск) за принятие дополнительных рисков. Суть модели заключается в том, чтобы определить эту дополнительную доходность, необходимую для компенсации рисков инвестиций.

$$R_i = R_f + \beta_i(R_m - R_f) \quad (2.14)$$

где :

R_i - доходность актива i

R_f - безрисковая ставка доходности

β_i - коэффициент актива i , который измеряет его волатильность относительно рынка

R_m - ожидаемая доходность рыночного портфеля

С развитием вычислительной техники стали применяться более сложные методы, включая линейное программирование [43], эволюционные алгоритмы [44], имитацию отжига [45], машинное обучение [46] и выпуклую оптимизацию. Последний подход рассмотрим подробнее.

Выпуклая оптимизация является одной из наиболее предпочтительных форм поиска оптимального портфеля благодаря некоторым свойствам. Одним из ключевых преимуществ выпуклой оптимизации является то, что любой локальный минимум выпуклой функции также является глобальным минимумом. Это свойство исключительно важно при оптимизации портфелей, так как позволяет инвесторам быть уверенными в том, что найденное решение является наилучшим возможным с точки зрения заданных критериев риска и доходности. Вывод о совпадении глобального и локального минимумов явно следует из определения выпуклости:

$$f(x) \text{ выпукла} \Leftrightarrow \forall x, y \in \text{dom } f, \forall \alpha \in [0; 1] \rightarrow f(\alpha x + (1 - \alpha)y) \leq \alpha f(x) + (1 - \alpha)f(y)$$

Выпуклые задачи, решаемые градиентными методами, имеют хорошую скорость сходимости. Так, классический градиентный спуск

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k) \quad (2.15)$$

сходится со скоростью $\mathcal{O}(\frac{1}{k})$, где k — номер итерации. Если функция также сильно выпукла (неравенство в формуле выше становится строгим), скорость сходимости улучшается до $\mathcal{O}(\exp(-k))$

Ускоренный метод Нестерова (Accelerated Gradient Descent)

$$\begin{aligned} y_k &= x_k + \beta_k(x_k - x_{k-1}) \\ x_{k+1} &= y_k - \alpha_k \nabla f(y_k) \end{aligned} \quad (2.16)$$

сходится для выпуклых функций со скоростью $\mathcal{O}(\frac{1}{k^2})$, что значительно быстрее по сравнению с классическим градиентным спуском.

Выпуклая оптимизация позволяет легко включать различные типы ограничений, такие как бюджетные ограничения, ограничения на максимальную или минимальную долю актива в портфеле, а также регуляторные ограничения. Это особенно важно для управления портфелями, где необходимо соблюдение определенных инвестиционных политик и норм.

Ещё одно преимущество выпуклой оптимизации заключается в том, что она позволяет моделировать широкий спектр целевых функций, таких как минимизация риска (например, дисперсии или среднеквадратичного отклонения) или максимизация доходности, а также комбинации обеих этих целей. Эта гибкость очень важна при построении портфелей, которые должны соответствовать конкретным предпочтениям и целям инвестора.

Stephen Boyd, решая задачу аллокации активов в портфеле [47] с помощью выпуклой оптимизации, рассматривает следующую постановку:

$$\begin{aligned} \text{Max}_{z_t} \quad & \hat{r}_t^T w_t + \hat{r}_t^T z_t - \phi_t^{trade}(z_t) - \phi_t^{hold}(w_t + z_t) - \gamma_t \cdot \psi(w_t + z_t) \\ \text{subject to} \quad & z_t \in \mathcal{Z}_t, w_t + z_t \in \mathcal{W}_t \\ & 1^T z_t + \phi_t^{trade}(z_t) + \phi_t^{hold}(w_t + z_t) = 0 \end{aligned} \quad (2.17)$$

где: \hat{r}_t^T - вектор ожидаемых доходностей по всем активам

w_t - веса текущего портфеля

z_t - вектор перераспределения весов портфеля

ϕ_t^{trade} - функция транзакционных издержек

ϕ_t^{hold} - функция издержек по заёму акций у брокера

γ_t - коэффициент принятия риска

$\psi(w_t + z_t)$ - функция оценки риска

Данная постановка является очень удобной, так как позволяет модернизировать задачу, подбирая функции издержек или риска индивидуально под себя или вовсе избавляться от них при необходимости. Важно учитывать, что данная задача может быть как выпуклой так и нет, в зависимости от выбранных функций издержек и оценки риска. Выпуклость - это важное условие, которое делает эту задачу комфортной в решении, поэтому выбор перечисленных функций является ответственным процессом.

В работе Miguel Sousa Lobo [48] рассматривается эвристический метод, который позволяет бороться с невыпуклостью некоторых задач. Этот метод субоптимального поиска портфеля основан на решении большого количества выпуклых задач вместо

одной невыпуклой. Таким образом, он возвращает субоптимальное решение, а также верхнюю границу оптимального решения. Численные эксперименты показывают, что невязка оптимального и субоптимального решения мала, даже когда размерность задачи(количество активов) превышает 100.

Более детальное обсуждение выбранного способа балансировки портфеля приводится в главе 3.

Глава 3

Практическая часть

3.1 Данные и их обработка

В качестве источника данных для проведения экспериментов было принято использовать Python-библиотеку `toexalgo`, которая официально разработана МосБиржей для участников рынка, занимающихся алгоритмическим трейдингом или тестирующими свои торговые алгоритмы на исторических данных. Среди всем преимуществ данной библиотеки следует выделить удобство использования, быстроту работы и доступ к 57 техническим индикаторам, которые можно использовать в качестве признакового описания ряда. Полный список индикаторов доступен в приложении А, в таблицах А.3, А.2 и А.1. В рамках эксперимента предполагается, что данные индикаторы позволяют модели улучшить свой прогноз цены на будущий период. Индикаторы можно сгруппировать на 3 класса:

1. Метрики, рассчитанные на потоке сделок: цены, объемы, соотношения покупок и продаж
2. Метрики, рассчитанные на потоке заявок: количество и объемы выставленных или снятых заявок
3. Метрики, рассчитанные на стакане котировок: количество уровней цен, спреды, ликвидность и дисбаланс покупок/продаж

Кроме того, в дополнительной серии экспериментов, в качестве признаков будут использоваться цены других активов. Данная постановка нацелена на проверку гипотезы о том, что разные активы связаны между собой, а значит между поведением их цен может существовать взаимосвязь, которая поможет при прогнозировании.

Перед началом обучения, в целях избежания влияния масштаба признаков, каждый из признаков, а также целевая переменная, прошли процесс стандартизации посредством `StandardScaller`, который определяется формулой

$$StandardScaller(X) = \frac{X - \mu}{\sigma^2} \quad (3.1)$$

где μ, σ – это среднее и дисперсия.

Также, во избежание проблем с пропущенными данными, все наблюдения содержащие пропуски были удалены из обучающей выборки.

3.2 Исследуемые модели

В рамках проводимого эксперимента, было принято решение использовать 4 архитектуры, которые будут обучаться с нуля. Используются такие модели, как LSTM, TSMixer, Transformer(Encoder), Transformer(Encoder+Decoder). Данные подходы зарекомендовали себя при работе с временными рядами и являются основополагающими подходами при решении задачи прогнозирования. Перечисленные модели обучались с нуля на тренировочной выборке и проверялись на валидационной.

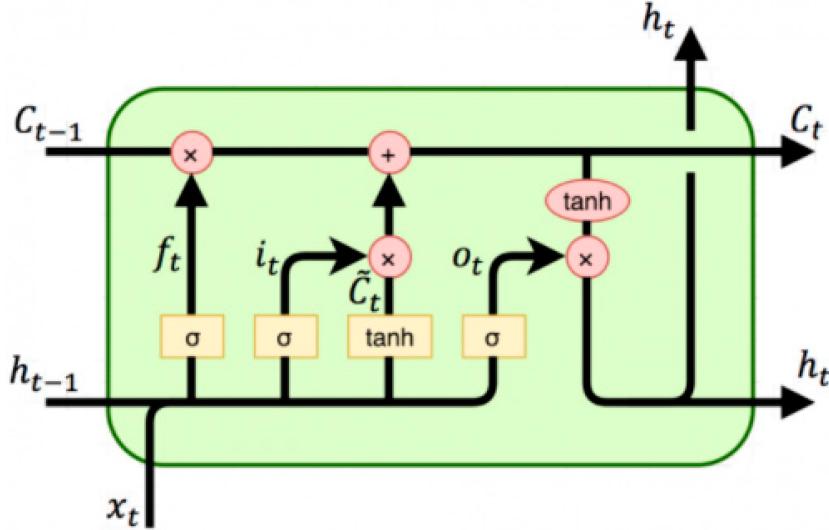


Рис. 3.1: LSTM архитектура

LSTM архитектура, нарисованная на рисунке 3.1 представляет из себя рекуррентную структуру, которая обрабатывает входящую последовательность шаг за шагом, вырабатывая при этом вектора скрытых состояний h_t , которые можно использовать для прогнозов. Данный подход умеет улавливать закономерности между данными благодаря своим каналам долгосрочной и краткосрочной памяти. Более формальное описание LSTM ячейки следующее:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (3.2)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (3.3)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \quad (3.4)$$

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \quad (3.5)$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (3.6)$$

$$h_t = o_t \cdot \tanh(C_t) \quad (3.7)$$

Как можно видеть из формул, LSTM представляет из себя комбинацию линейных преобразований с использованием функций активации сигмоиды и гиперболического тангенса. Данные, поступающие в ячейку, разделяются, попадая в канал долгосрочной памяти C_t и краткосрочной памяти h_t , в свою очередь веса LSTM ячейки

обучаются таким образом, что бы хранить в C_t ту информацию, которую требуется сохранять на протяжении всей работы блока. Так или иначе, как и все рекуррентные архитектуры, LSTM имеет такие проблемы, как ограниченные возможности запоминания(хотя и увеличенные по сравнению с обычной RNN), сложность вычисления градиентов из-за своей рекуррентной структуры а также затухание градиентов.

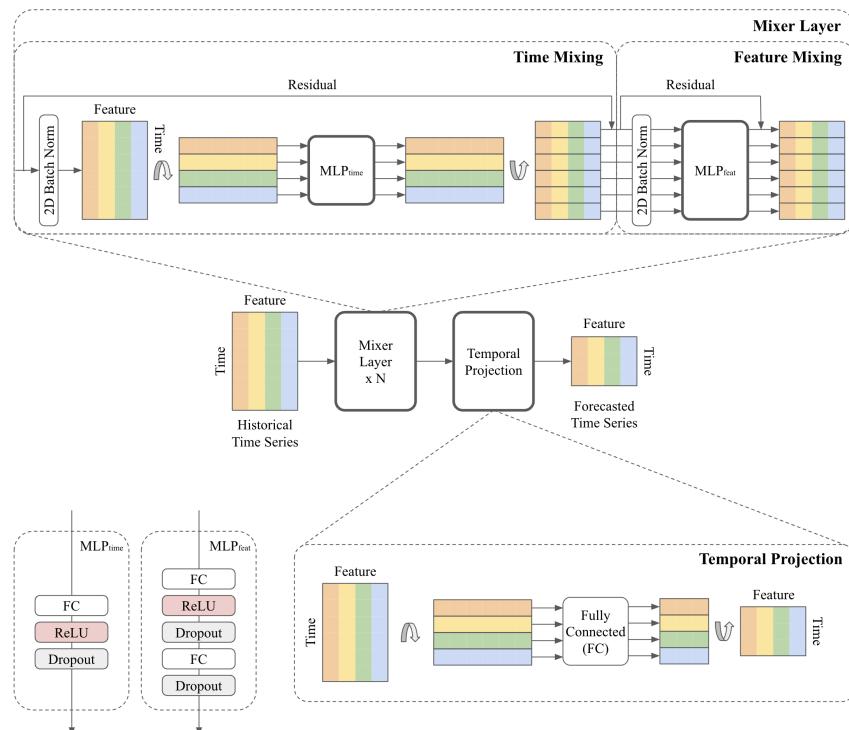


Рис. 3.2: TSMixer архитектура

TSMixer архитектура, нарисованная на рисунке 3.2 представляет из себя комбинацию из линейных слоёв, которые поочерёдно обрабатывают входящие данные. На первом этапе, входящая матрица размера (num observations, num features) транспонируется и проходит сквозь полно связанный слой, с целью определить авторегрессионные паттерны в данных, так как возможно, некоторые значения цены и признаков зависят от своих же значений несколько шагов назад. На следующем этапе, полученная матрица снова транспонируется в исходное состояние и прогоняется через другой полно связанный слой, задачей которого является определение зависимостей между ценой актива и техническими индикаторами. Кроме того, в данной архитектуре реализованы такие техники как ResidualConnection, Dropout и BatchNorm для поддержания качественного обучения. Важно отметить, что данная архитектура предназначена исключительно для многомерных временных рядов, а значит не будет проходить проверку на серии испытаний с одномерным рядом без каких либо признаков.

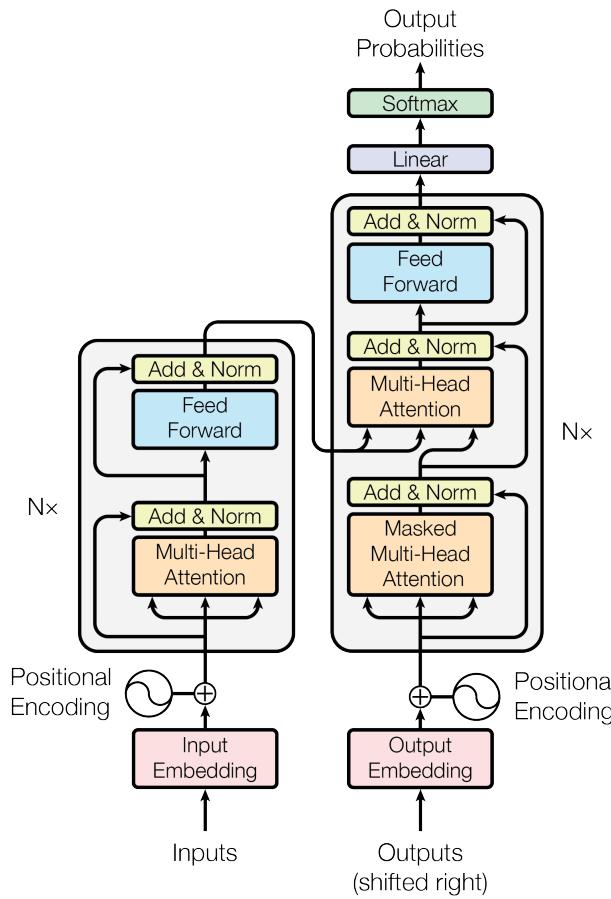


Рис. 3.3: Transformer архитектура

Представленная на рисунке 3.3 архитектура Transformer имеет две основные части: энкодер(левая часть) и декодер(правая часть). Входящие в энкодер данные предварительно проходят через Embedding слой, представляющий из себя полно-связный слой, создающий вектора-эмбеддинги исходного наблюдения. После прохождения данного слоя, наблюдения попадают в PositionalEncoding слой, который добавляет информацию об очерёдности наблюдений и создаёт структуру входящей последовательности.

Существует два варианта реализации слоя, предложенные в оригинальной статье [10]:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d}}}\right) \quad (3.8)$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i}{d}}}\right) \quad (3.9)$$

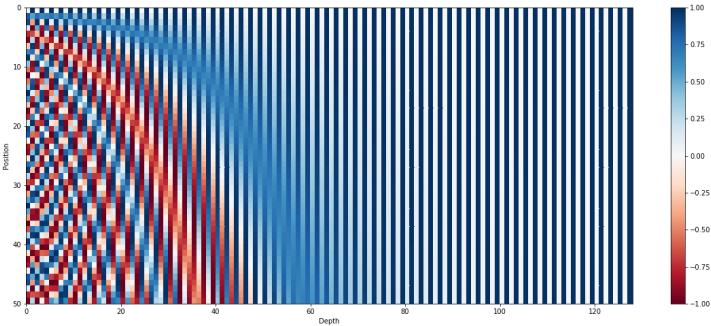


Рис. 3.4: Паттерн PositionalEncoding слоя

Обе функции создают матрицу, имеющую паттерн, позволяющий модели определять позиции входящих в последовательность наблюдений. Эмпирические эксперименты показали, что данный слой улучшает качество работы модели [10].

После прохождения данного слоя, входящие данные попадают в Multi-Head Attention слой, внутри которого происходит установление зависимостей между входящими данными.

$$MultiHead(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O \quad (3.10)$$

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \quad (3.11)$$

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (3.12)$$

Затем последовательность поступает в полносвязный слой с дальнейшим выходом, размерность которого равна длине прогнозируемого периода.

Декодер-блок устроен по схожему принципу, однако использует маску, запрещающую модели устанавливать взаимосвязи между векторами на позициях i и j , если $i < j$. Это ограничение позволяет избежать рекуррентного вычисления следующих значений.

Как говорилось ранее, в рамках проводимых экспериментов используются две вариации модели: первая содержит исключительно энкодер, а вторая и энкодер и декодер. Обе конфигурации модели уместны в рамках постановки задачи.

3.3 Функция потерь

Выбор правильной функции потерь является одним из ключевых этапов при решении задачи прогнозирования временного ряда и требует рассмотрения некоторых особенностей самих данных и целей их прогнозирования. Как упоминалось в предыдущей главе, временные ряды бывают разные и цели их прогнозирования, соответственно, тоже. Так, например, метеорологи прогнозируют будущую температуру для составления прогноза погоды и погрешность прогноза в n градусов одинаково страшна как в случае, когда прогноз выше ожидаемой температуры, так и когда прогноз ниже её на те же n градусов. Исходя из этого, исследователи хотят, что бы прогноз приближался к истинному значению и поэтому в качестве Loss-функций могут быть использованы такие функции потерь как MSE (Среднеквадратичная ошибка), MAE (Средняя абсолютная ошибка) и иные симметричные функции. Под симметричными функциями предполагается такой класс функций, для которых значение от аргументов x и $-x$ одинаково. Более формально:

$$f : \mathbb{R}^n \rightarrow \mathbb{R} \text{ симметрична} \iff \forall x \in \text{dom } f \rightarrow f(x) = f(-x)$$

Данный класс функций широко используется в задачах прогнозирования рядов, когда знак ошибки не имеет значения. В случае прогнозирования финансовых данных, у исследователя существует спекулятивный интерес, который он пытается удовлетворить используя прогнозы ряда. Говоря о прогнозировании цен акций, человек прогнозирует их будущую цену с целью совершения транзакций для извлечения прибыли. Это означает, что для него знак ошибки может играть ключевую роль, как было показано в обзорной части работы.

В связи с этим, в работе предлагается модифицировать функцию MAPE путём введения штрафа за неугаданный знак доходности. Используя предложения из работы [27] и адаптируя их под текущую постановку, была получена модифицированная MAPE функция со штрафом за неугаданный знак:

$$CustomLoss(y_{i+1}, \hat{y}_{i+1}) = \left| \frac{y_{i+1} - \hat{y}_{i+1}}{y_{i+1}} \right| - \lambda(\hat{y}_{i+1} - y_i)(y_{i+1} - y_i) \quad (3.13)$$

Как можно видеть из формулы, часть $\left| \frac{y_{i+1} - \hat{y}_{i+1}}{y_{i+1}} \right|$ полностью соответствует функции MAPE, однако далее идёт добавленный штраф с коэффициентом чувствительности λ . Если знаки $(\hat{y}_{i+1} - y_i)$ и $(y_{i+1} - y_i)$ совпадают, то знак этого произведения всегда будет положительный, а значит коэффициент λ должен идти со знаком минус. И наоборот, если члены произведения отличаются знаками, то предсказанное направление движения актива отличается от его истинного движения, а следовательно это должно увеличивать функцию потерь. Коэффициент λ , в свою очередь, отвечает за величину штрафа и является гиперпараметром. При слишком большом λ модель будет стараться угадывать направление в ущерб точности самого прогноза и наоборот, при слишком маленьком λ модель не будет бояться ошибаться в направлениях. Следовательно, этот коэффициент должен быть таким, чтобы модель при обучении соплась к весам, с которыми могла бы угадывать направление, однако делала это с приемлемой точностью.

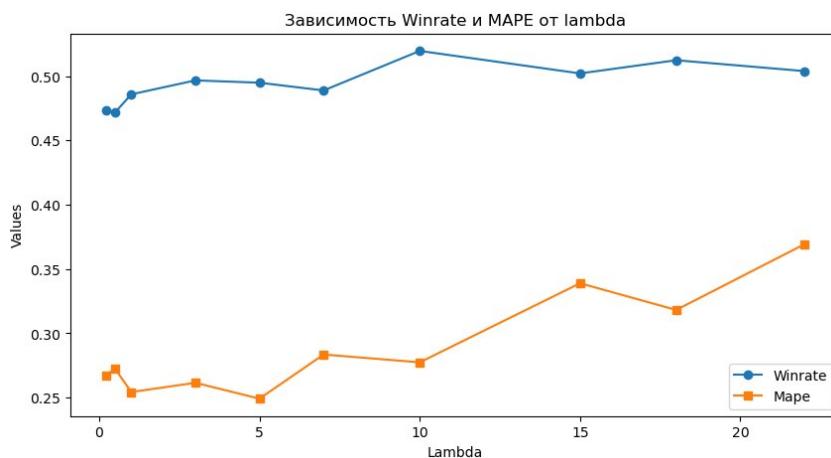


Рис. 3.5: Зависимость Winrate и MAPE от гиперпараметра λ

Эмпирические исследования подтверждают теоретическое представление о работе штрафной компоненты функции. Как видно на рисунке 3.5, метрика Winrate (доля правильно угаданных направлений) имеет тенденцию расти при росте гиперпараметра λ , и, как следствие, это влечёт за собой динамику метрики MAPE, которая также растёт. Очевидно, не существует единственно правильного ответа на вопрос о том, какое значение параметра выбрать для максимизации качества прогнозов. С одной стороны, мы хотим получать достоверные прогнозы с точки зрения их точности, так

как если бы мы старались максимизировать Winrate путём сильного увеличения λ , то модель хоть и начинала бы чаще угадывать направление, однако из-за низкой точности цена ошибки была бы высока. И наоборот, стараясь минимизировать MAPE путём понижения λ , мы рискуем получать низкую частоту угадываний направления, и, как следствие, недостаточный Winrate для ведения прибыльной торговли. Поэтому, раз λ служит параметром, который находит компромисс, то наилучшее его значение должно искаться в ходе итеративного перебора.

3.4 Поиск оптимального портфеля

Перед началом обсуждения итоговой методологии эксперимента, следует уделить внимание методам поиска оптимального портфеля для тестирования качества прогнозов. В рамках данного эксперимента, наши модели прогнозируют будущую цену актива, а значит, из ожидаемой цены возможно получить ожидаемую доходность путём следующей операции:

$$\hat{r}_{i+1} = \frac{\hat{y}_{i+1} - y_i}{y_i} \quad (3.14)$$

где y_i известна, а \hat{y}_{i+1} это ожидаемая цена

Из этого следует, что в каждый момент времени, при условии прогнозирования N активов на рынке, мы можем получить вектор размерности N , каждой координатой которого будет являться прогнозируемая доходность определённого актива на следующий момент времени. Данный вектор позволяет нам переходить к задаче оптимизации портфеля, где целевой переменной может выступать вектор весов портфеля, вектор вложений портфеля, вектор торговых операций и так далее.

Используемая в ходе эксперимента задача оптимизации является адаптацией задачи, предложенной Стивеном Бойдом в работе [47] и представляет из себя однопериодную оптимизацию портфеля. Однопериодная задача оптимизации инвестиционного портфеля заключается в правильном выборе распределения средств, которые направляются на покупку и продажу активов. Данная задача решается путём оптимизации выпуклой функции градиентными методами. В рамках статьи [47], авторы рассматривают в качестве целевой переменной не сам портфель, а вектор торговых операций, что позволяет легче решать задачу округления под рыночные цены, так как изначально задача решается в предположении о бесконечной делимости акций, что на практике не выполняется.

Проведём более детальное рассмотрение задачи оптимизации путём изучения её составных частей:

1. $h_t \in \mathbb{R}^{n+1}$ инвестиционный портфель с n активами в момент времени t , где $(h_t)_{n+1}$ это денежный баланс. В оригинальной статье данная координата называется Cash Account, что может быть проинтерпретировано как брокерский счёт. Это те средства, которые лежат у нас на балансе, но не задействованы на данный момент.
2. $p_t \in \mathbb{R}_+^n$ - вектор цен на активы в начале периода t .
3. $v_t = 1^T h_t$ - Net Asset Value(NAV). Стоимость портфеля в момент t .
4. $w_t \in \mathbb{R}^{n+1}$ - веса, связанные с распределением активов в портфеле, определяемые через $w_t = h_t/v_t$ и имеющие свойство $1^T w_t = 1$. $(w_t)_{n+1}$ это доля общей стоимости портфеля, лежащей на брокерском счету.

5. $u_t \in \mathbb{R}^n$ - выраженный в рублях вектор торговых операций. $(u_t)_i > 0$ означает покупку актива i , $(u_t)_i < 0$ означает продажу актива i . $(u_t)_{n+1}$ означает сумму, которая идёт на брокерский счёт.

6. $z_t = u_t/v_t$ - отнормированный на стоимость портфеля вектор операций u_t

7. $h_t^+ \in \mathbb{R}^{n+1}$ - портфель после операций, проведённых в начале периода t (Post-trade portfolio). $h_t^+ = h_t + u_t \forall t \in \{1, 2, 3, \dots, T\}$. Отсюда следует, что:

$$v_t^+ = 1^T h_t^+$$

$$v_t^+ - v_t = 1^T h_t^+ - 1^T h_t = 1^T u_t$$

8. $w_t + z_t = h_t^+/v_t$ - отнормированный портфель после торговых операций (следует из того, что $h_t^+ = h_t + u_t$)

9. $\phi_t^{trade}(u_t) : \mathbb{R}^{n+1} \rightarrow \mathbb{R}$ - функция транзакционных издержек, обладающая свойствами:

$$\phi_t^{trade}((u_t)_{n+1}) = 0$$

$$\phi_t^{trade}(x) = \sum_{i=1,..,n} (\phi_t^{trade})_i(x_i)$$

10. $\phi_t^{hold}(h_t + u_t) : \mathbb{R}^{n+1} \rightarrow \mathbb{R}$ - функция издержек займа акций для короткой позиции:

$$\phi_t^{hold}(h_t + u_t) = s_t^T (h_t + u_t)_-$$

где $(x)_- = \max\{-x, 0\}$

$$\phi_t^{hold}(h_t + u_t)_{n+1} = 0$$

11. $1^T u_t + \phi_t^{trade}(u_t) + \phi_t^{hold}(h_t + u_t) = 0$ - условие самофинансирования, говорящее о том, что для проведения покупок/продаж на рынке издержки на транзакции и удержание активов должны быть гарантированно покрыты. Отсюда следует, что:

$$(u_t)_{n+1} = -(1^T (u_t)_{1:n} + \phi_t^{trade}((h_t + u_t)_{1:n}) + \phi_t^{hold}((u_t)_{1:n}))$$

12. $(r_t)_i = \frac{(p_{t+1})_i - (p_t)_i}{(p_t)_i}$ - доходность актива i .

13. $h_{t+1} = h_t^+ + r_t \circ h_t^+ = (1 + r_t) \circ h_t^+$ - портфель на этапе $t+1$. Где \circ - произведение Адамара.

14. $v_{t+1} = v_t + r_t^T h_t + t_t^T u_t - \phi_t^{trade}(u_t) - \phi_t^{hold}(h_t^+)$ - стоимость портфеля в $t+1$ момент времени.

15. $R_t^p = r_t^T w_t + r_t^T z_t - \phi_t^{trade}(z_t) - \phi_t^{hold}(w_t + z_t)$ - доходность портфеля за период t

Последний 14й пункт позволяет нам сформулировать целевую функцию, которую мы будем оптимизировать.

Общая постановка задачи:

$$\begin{aligned} \text{Max}_{u_t} \quad & \hat{r}_t^T h_t + \hat{r}_t^T u_t - \phi_t^{trade}(u_t) - \phi_t^{hold}(h_t + u_t) - \gamma_t \cdot \psi(h_t + u_t) \\ \text{subject to} \quad & u_t \in \mathcal{U}_t, \quad h_t + u_t \in \mathcal{W}_t \\ & 1^T u_t + \phi_t^{trade}(u_t) + \phi_t^{hold}(h_t + u_t) = 0 \end{aligned}$$

где : $\psi_t : \mathbb{R}^{n+1} \rightarrow \mathbb{R}$ функция неприятия риска
 $\gamma_t > 0$ Параметр чувствительности

Используя множества $\mathcal{U}_t, \mathcal{W}_t$ мы можем моделировать поведение алгоритма. Так, например, $h_t + u_t \geq 0$ означает, что алгоритму запрещено занимать короткие позиции, так как каждая координата post-trade portfolio должна быть ≥ 0 .

Оценка сложности:

Задача однопериодной оптимизации может быть решена методами внутренней точки со сложностью $\mathcal{O}(nk^2)$ где n это число активов, а k это число факторов(при факторной модели оценки риска) и за $\mathcal{O}(n^3)$ при других методах оценки риска.

Существует особый случай, который может быть решен гораздо быстрее. Если целевая функция квадратична, т.е. риск и издержки являются квадратичными функциями, а единственными ограничениями являются линейные ограничения-равенства, то задача может быть решена со сложностью $\mathcal{O}(nk^2)$, что примерно в 50 раз быстрее чем использование метода внутренней точки.

Для перехода к частной постановке задачи, требуется определение функций транзакционных издержек, комиссий брокеру за заём акций для короткой позиции и определение функции риска. В рамках эксперимента, алгоритму запрещено занимать короткие позиции, а значит $\phi_t^{hold}(h_t + u_t)$ отсутствует. $\phi_t^{trade}(u_t)$ в свою очередь представляет из себя скалярное произведение $c^T u_t$, где c - вектор констант, равных комиссиям биржи и брокера, а $\psi(h_t + u_t)$ представляет из себя квадратичную форму $(h_t + u_t)\Sigma(h_t + u_t)$, где Σ - ковариационная матрица доходностей активов.

Частная постановка задачи:

$$\begin{aligned} \text{Max}_{u_t} \quad & \hat{r}_t^T h_t + \hat{r}_t^T u_t - c^T u_t - \gamma_t \cdot ((h_t + u_t)\Sigma(h_t + u_t)) \\ \text{subject to} \quad & u_t \in \mathbb{R}^{n+1}, \quad h_t + u_t \geq 0 \\ & 1^T u_t + c^T u_t = 0 \end{aligned}$$

где : c - вектор комиссий брокеру и бирже
 h_t - портфель до торговых операций
 u_t - вектор торговых операций
 \hat{r}_t - ожидаемые доходности
 γ_t - коэффициент принятия риска(гиперпараметр)
 Σ - ковариационная матрица доходностей

Решение данной задачи производилось с использованием библиотеки CVXPY, написанной на языке программирования Python. Эксперимент производился для 42 компаний из индекса Московской Биржи, следовательно, с учётом дополнительной координаты брокерского счёта, размерность задачи составила 43. Начальной точкой являлся вектор, состоящий из нулей по первым 42 координатам и 1.000.000 по 43 координате. Иными словами, мы начинаем с пустого портфеля и одного миллиона на брокерском счёте.

При решении задачи используется пакет CVXPY и солвер SCS.

Splitting Conic Solver (SCS) является итеративным солвером для выпуклых задач оптимизации, основанным на методе разложения Дугласа-Рачфорда и его вариациях.

Это метод подходит для решения задач конической оптимизации и работает путём итеративного разложения задачи на более простые подзадачи.

Метод Дугласа-Рачфорда — это итеративный метод разложения, использующий проксимальные операторы для решения выпуклых задач оптимизации. SCS может решать линейные задачи (LP), задачи квадратичной оптимизации (QP), задачи второго косинусного порядка (SOCP), полуопределённое программирование (SDP) и задачи конусной оптимизации. Алгоритм состоит из следующих шагов:

1. Разложение задачи на подзадачи: весь процесс начинается с разбиения исходной задачи на более простые подзадачи
2. Применение проксимальных операторов к каждой подзадаче: каждой подзадаче применяется соответствующий проксимальный оператор. Проксимальный оператор для заданной функции f находит точку, которая минимизирует f с добавлением квадратичного штрафа за отклонение от текущего решения
3. Комбинирование результатов для обновления решения: результаты, полученные из подзадач, комбинируются для обновления текущего решения задачи. На этом этапе информация из всех подзадач объединяется, чтобы приблизиться к решению исходной задачи

Теоретические оценки:

SCS имеет доказанные теоретические результаты по скорости сходимости. Основные результаты включают:

1. Линейная скорость сходимости: Для задач, где конус K является гладким и сильно выпуклым, метод Дугласа-Рачфорда показывает линейную скорость сходимости.
2. Скорость сходимости для общих задач: Для общих выпуклых задач скорость сходимости SCS зависит от свойств проксимальных операторов и структуры задачи. Обычно SCS показывает сублинейную скорость сходимости, что характерно для большинства методов первого порядка.
3. Анализ сложности: SCS демонстрирует, что количество итераций для достижения решения с точностью ϵ для задач с гладкими функциями и сильно выпуклыми конусами оценивается как $O(\log(1/\epsilon))$. Для общих задач количество итераций оценивается как $O(1/\epsilon)$ где ϵ это невязка.

Скорость на задачах разной размерности:

При размерности задачи в 43 актива, среднее время выполнения составляет $0.02sec \pm 0.0016sec$

При размерности задачи в 430 активов, среднее время выполнения составляет $0.23sec \pm 0.0019sec$

При размерности задачи в 840 актива, среднее время выполнения составляет $1.21sec \pm 0.0277sec$

При размерности задачи в 1600 актива, среднее время выполнения составляет $8.81sec \pm 0.1216sec$

Как можно видеть, скорость выполнения одной итерации сильно возрастает, когда число активов стремится к тысяче, однако в рамках нашей постановки с размерностью 43(42 актива + брокерский счёт), солвер показывает приемлемое время и может быть использован без опасений о крупных временных задержках.

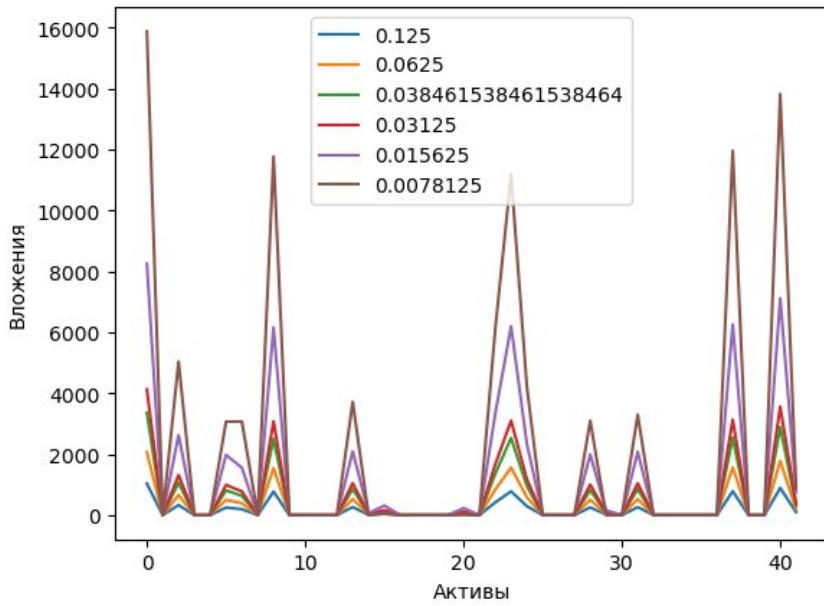


Рис. 3.6: Типичное решение задачи

На рисунке 3.6 изображена зависимость типичного решения задачи от параметра γ . Полученные результаты соотносятся с действительностью, так как чем меньше трейдер реагирует на риск, тем более агрессивную торговлю он ведёт. Вдоль оси X отложены активы, а вдоль оси Y отложена величина средств, вкладываемая в актив. Таким образом, чем ниже параметр чувствительности к риску, тем больше средств алгоритм вкладывает в покупку активов.

Дополнительно была проведена проверка соответствия точки решения всем ограничениям, которые накладывались при постановке. В результате проверки выяснилось, что все ограничения соблюдены, однако в нулевых координатах иногда встречается шум, размера не больше чем $1e^{-6}$. Такой же шум наблюдается при проверке соблюдения условия самофинансирования $1^T u_t + c^T u_t \approx 1e^{-6}$.

3.5 Методология эксперимента

Составные части эксперимента, которые обсуждались выше, позволяют нам перейти к полному описанию методологии проводимого исследования.

Данные:

2020 2021 2022(с 1 марта) 2023 2024(по 1 апреля)

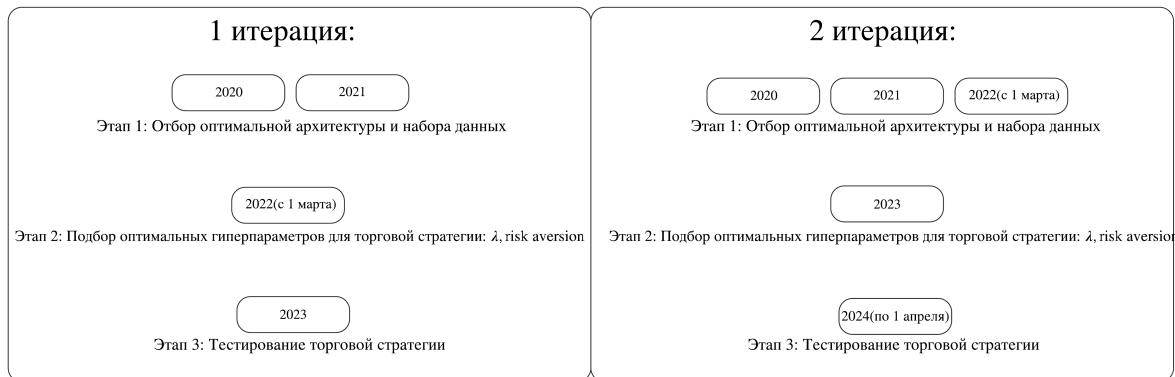


Рис. 3.7: Схема проведения эксперимента

Как видно из рисунка 3.7, данное исследование можно разбить на две части, каждая из которых содержит три основных этапа:

1. Поиск оптимальной архитектуры и набора данных
2. Оптимизация торговой стратегии по гиперпараметрам
3. Тестирование торговой стратегии

Имея данные с 2020 по апрель 2024 года, было принято решение разбить эксперимент на две итерации. Целью такого разбиения является проверка адаптивности стратегии к изменению обучающих данных. Так, в первой итерации подбор оптимальной архитектуры производится на данных 2020-2021 годов, а на второй итерации это происходит в том числе и на части данных 2022 года. Аналогично первому этапу, данные второго и третьего этапов также отличаются от итерации к итерации, что позволяет нам оценить изменения в торговой стратегии по мере изменения конъюнктуры рынка.

3.5.1 Поиск оптимальной архитектуры и набор данных

Задачей данного этапа является нахождение оптимальной архитектуры и постановки задачи, которые обеспечивают наименьшее значение функции потерь на разных значениях λ . Сравнивая модели по данному показателю, мы отвечаем на вопрос о том, какая из архитектур способна наилучшим образом находить внутренние взаимосвязи во входящей последовательности и производить наиболее качественное прогнозирование как с точки зрения MAPE, так и с точки зрения Winrate.

Таблица 3.1: Таблица испытаний

	И.1	И.2	И.3
LSTM	•	•	•
Transformer(encoder часть)	•	•	•
Transformer(encoder + decoder)	•	•	•
TSMixer	•	•	•

Где:

И.1: Одномерный ряд(без признаков), прогноз на 1 час вперёд.

И.2: Многомерный ряд(признаки - технические индикаторы), прогноз на 1 час вперёд.

И.3: Многомерный ряд(признаки - другие активы на рынке), прогноз на 1 час вперёд.

Этап представляет из себя серию из 11 испытаний. В ходе испытания, каждая из моделей будет обучаться и валидироваться на введённой ранее функции потерь с разными гиперпараметрами λ , затем для каждой модели и для всех значений λ

на которых она обучалась в рамках испытания, будет происходить усреднение по значению функции потерь на валидации. Процедура проходит на данных 2020 и 2021 годов с отведением 80 процентов выборки на обучение и 20 процентов выборки на валидацию в рамках первой итерации, а затем на данных 2020, 2021 и 2022 годов в рамках второй итерации.

Данный подход избегает необходимости сравнения моделей по метрикам Winrate и MAPE отдельно, так как в данном случае неизвестно, какая модель лучше, с высоким Winrate или низким MAPE. Лосс функция, в свою очередь, позволяет сравнивать архитектуры строго по своему значению: чем ниже лосс для фиксированного λ , тем лучше она умеет обобщать данные при выбранном гиперпараметре. После проведения эксперимента, мы получаем таблицу с усреднёнными результатами испытаний для $\lambda \in [0,1,5,10,12,25,35,45,50]$, а затем, анализируя полученные результаты, находим оптимальную модель и признаковое описание.

3.5.2 Поиск оптимальных гиперпараметров торговой стратегии

Получив оптимальную модель и признаковое описание на предыдущем шаге, происходит процесс её внедрения в торговый алгоритм.

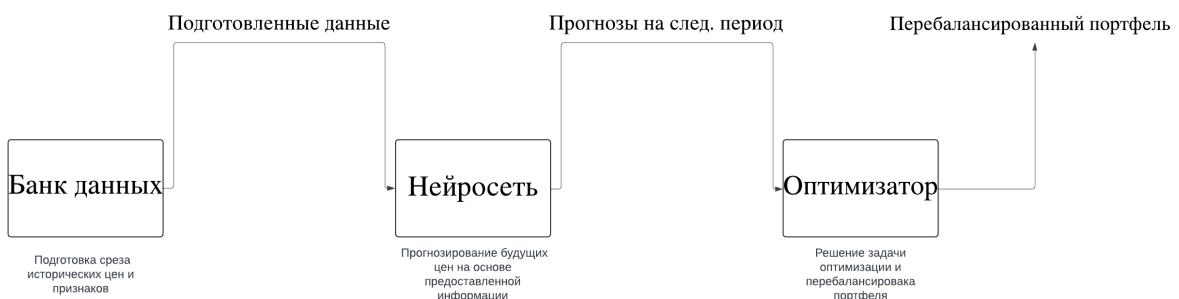


Рис. 3.8: Схема одной итерации алгоритма

На рисунке 3.8 визуализирована схема одной итерации алгоритма. Имея исторические данные цен и признаков, алгоритм генерирует для каждого актива входную матрицу размера (длина истории, количество признаков), это и есть исторические данные, подающиеся в нейросеть.

На следующем этапе алгоритма, интегрированная с прошлого шага нейронная сеть производит обработку исторических данных и генерирует прогнозы цен активов на следующий час.

Получив часовые прогнозы цен для всех активов, алгоритм подаёт их в блок, имеющий условное название "Оптимизатор". Внутри себя, "Оптимизатор" из прогнозов цен получает прогнозы доходностей, а затем, формируя из них вектор \hat{r} , решает оптимизационную задачу 3.4. Решением оптимизационной задачи является вектор торговых операций, который затем округляется по актуальным ценам. Наконец, "Оптимизатор" совершает торговые операции на рынке, тем самым перебалансируя портфель.

Формальное описание алгоритма приведено ниже:

Algorithm 1 Торговый алгоритм

Require:

$$h_0 = (0, 0, \dots, \text{initial balance}), c, \gamma, \Sigma, T ;$$

Ensure:

h_T - итоговый портфель после торговой сессии

- 1: **for** $t = 0 : T$ **do**
 - 2: Формируем исторические данные для подачи в нейросеть
 - 3: Прогнозируем цены активов к концу периода t
 - 4: Получаем из них прогнозы доходностей \hat{r}_t
 - 5: Решаем задачу оптимизации $u_t^* = \arg \max_u R_t^p(u)$
 - 6: Округляем u_t^* по ценам на момент t
 - 7: Совершаем торговые операции $h_t^* = h_t + u_t^*$
 - 8: Дожидаемся конца периода и фиксируем реализованную доходность $h_{t+1} = (1 + r_t) \circ h_t^*$
 - 9: **end for**=0
-

Старт работы алгоритма происходит в начале торгового сеанса путём инициализации стартового портфеля $h_0 = (0, 0, \dots, \text{стартовый баланс})$, значения транзакционных издержек c , коэффициента чувствительности к риску γ , ковариационной матрицы Σ и периода окончания торгового сеанса T . Далее запускается итеративный алгоритм, который ежесчасно выполняет описанные действия.

После периода T , алгоритм завершает свою работу и мы фиксируем доходность портфеля, а также его коэффициент Шарпа. Проход данного алгоритма совершается на данных 2022 года в рамках первой итерации, затем на данных 2023 года в рамках второй итерации.

Таким образом, для одной фиксированной связки (λ, γ) мы проводим один торговый сеанс на наших данных и фиксируем полученную метрику качества. Гиперпараметры могут принимать значения $\lambda \in [0, 1, 5, 10, 12, 25, 35, 45, 50]$ и $\gamma \in [0.5, 1, 2, 3, 4, 5, 8]$, а метрикой качества является коэффициент Шарпа. Выбор данной метрики основывается на необходимости оценки не только доходности портфеля, но и риска, возникающего при получении доходности. Данный коэффициент имеет формулу $S = \frac{R_p - R_f}{\sigma_p}$, где σ_p - это стандартное отклонение доходности портфеля, отражающее в себе риск.

Совершив 63 прохода алгоритмом, необходимо выбрать ту комбинацию (λ, γ) , которая максимизирует коэффициент Шарпа.

3.5.3 Тестирование полученной стратегии

После проведения первых двух пунктов, мы имеем:

1. Подобранную и обученную архитектуру
2. Оптимизированную торговую стратегию, которую необходимо протестировать

Совершая тестирование стратегии, необходимо решать эту задачу на данных, которые ещё не видели ни модель во время обучения, ни стратегия во время оптимизации[49].

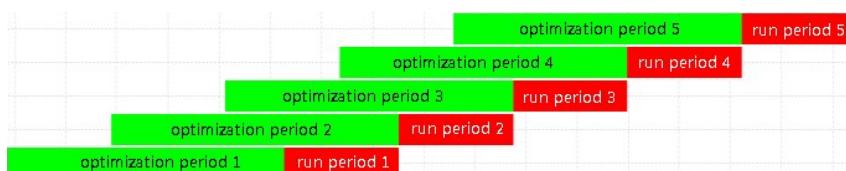


Рис. 3.9: Подход к оптимизации и тестированию торговой стратегии

Данный подход позволяет провести наиболее честную проверку модели, так как гарантирует факт отсутствия переобучения под тестовые данные. Следуя схеме на рисунке 3.9, тестирование будет производиться на данных 2023 года в рамках первой итерации и на данных 2024 года(по 1 апреля) в рамках второй итерации. Тестирование представляет из себя процесс торговли с оптимальными λ и γ , полученными с предыдущего шага. После прохождения тестирования, фиксируются данные о доходности портфеля, его коэффициента Шарпа, исторические данные об изменении стоимости портфеля во время тестирования, а также другая вспомогательная информация. В рамках данного шага, мы смотрим на доходность портфеля, а также сравниваем её с доходностью бенчмарка(Индекс Мосбиржи) за аналогичный период.

Итого, после проведения обеих трехшаговых итераций, мы получим значения λ^* , γ^* для каждой из них, метрики качества в виде коэффициента Шарпа, а также доходности стратегий на тестовых множествах. После чего, будет проведён анализ и интерпретация полученных результатов, сравнение их с результатами бенчмарка (Индекс Мосбиржи), а также будут сделаны выводы об эффективности стратегий и перспектив их применения.

Глава 4

Результаты работы

Данная глава представляет из себя обзор и интерпретацию результатов, полученных в ходе проведённых экспериментов. Помимо этого, в рамках данной главы мы обсудим конкретную спецификацию исследуемых моделей, а также все технические детали проводимого эксперимента.

4.1 Результаты поиска оптимальной архитектуры

Напомним, что ключевой задачей нашего исследования на данном этапе является поиск оптимальной архитектуры, которая позволила бы наиболее качественно аппроксимировать временной ряд, а также наиболее точно предсказывать его будущие значения. В ходе этапа, четыре архитектуры проходят серию из трёх испытаний, единственным отличием которых является признаковое описание, используемое моделью.

Так, первое испытание (И.1) представляет из себя прогнозирование цен закрытий без каких либо признаков. Второе испытание (И.2) представляет из себя такое же прогнозирование цены закрытия, однако с использованием технических индикаторов в качестве признаков. Предполагается, что данные индикаторы содержат скрытые сигналы, используя которые возможно определить будущее поведение цены. Третье испытание (И.3), аналогично И.2, помимо цены использует признаковое описание ряда, однако в этот раз в качестве признаков выступают цены других активов, торгующихся на Московской Бирже. Предполагается, что некоторые активы взаимосвязаны между собой и их цены имеют ненулевую корреляцию. Как правило, такие активы принято разбивать на кластеры по разным признакам[50]. В качестве признака для создания кластера может быть использован как признак сектора или отрасли, так и волатильность инструмента или рейтинговая оценка компании эмитента. В рамках нашей работы мы не выделяем определённых признаков и не занимаемся кластеризацией, однако проверяем, сможет ли нейросеть вычислить некоторые паттерны во взаимосвязи не только цены целевого актива между собой, но и взаимосвязи цены целевого актива с ценами других активов на рынке.

Как было сказано ранее, наш эксперимент состоит из двух итераций (каждая по три этапа), отличающихся лишь набором используемых данных. Следовательно, набор исследуемых моделей, их спецификация, признаковое описание и методы предобработки данных остаются неизменными, а сами наборы данных изменяются. В рамках данной секции обсуждаются общие принципы работы с данными, используемые в эксперименте.

В качестве источника данных, в работе использовалась библиотека MoexAlgo, разработанная официально Московской Биржей на языке программирования Python.

Данная библиотека формирует внутренние запросы с помощью API Московской Биржи и предоставляет данные пользователю в виде файлов с расширением '.csv'. Поскольку источником данных является сама биржа, то в ходе эксперимента не возникает опасений по поводу их достоверности.

В рамках проводимого эксперимента, для нормализации данных использовался метод StandardScaler от библиотеки Sklearn. Данный подход представляет из себя почленное вычитание среднего и деление на дисперсию каждого члена временного ряда. В итоге, после проведения процедуры, распределение значений цены временного ряда, а также каждого из признаков (если таковые имеются) описывается нормальным распределением $\mathcal{N}(0,1)$. Важно отметить следующие особенности обработки:

Помимо цены актива, мы имеем 57 технических индикаторов, которые также необходимо обработать в целях повышения качества и стабильности обучения. Представим нашу выборку в следующем виде:

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots \\ \vdots & \ddots & \\ a_{N1} & & a_{NM} \end{bmatrix}$$

Здесь N = количеству наблюдений, M = 58(целевая переменная + 57) в рамках испытания 2, или M = 42(целевой актив и 41 актив из индекса) в рамках испытания 3 или M = 1(только целевая переменная) в рамках испытания 1. Следовательно, для каждой из колонок матрицы должен существовать свой StandardScaler, который рассматривает каждую колонку индивидуально и обрабатывает её, используя μ, σ вычисленные по значениям из этой колонки. Кроме того, поскольку 20 процентов выборки уходит на валидацию, каждый из StandardScaller'ов должен обучаться на 80 процентах наблюдений, а затем, на основе обученных μ, σ , трансформировать и тренировочную и валидационную подвыборки. Иными словами, scaler.fit делается на тренировочной выборке($N*0.8$), а затем scaler.transform делается на тренировочной($N*0.8$) и валидационной($N*0.2$). Данный подход позволяет избежать утечки данных, так как если бы мы тренировали μ, σ на всей выборке, то часть данных из тренировочной части попала бы в валидационную и наоборот.

Кроме того, Scaler, обрабатывающий целевую переменную должен быть сохранён для проведения процедуры обратной трансформации предсказанных на валидационной выборке данных, в то время как остальные Scaler'ы больше не понадобятся. Для удобства, назовём необходимый к сохранению Scaler как Scaler*. Итого, после проведения обработки данных, мы имеем матрицу A_{NM} и Scaler*, который натренирован на целевой переменной из A .

После нормализации данных, наступает черёд приведения данных к формату, требуемому для подачи их в нейронную сеть. Нарезание данных происходит техникой бегущего окна с фиксированным размером, равным 32 в случае нашего эксперимента. Для примера возьмём случай испытания 2, где матрица данных = $A_{N,58}$ и предположим, что первой колонкой матрицы A является целевая переменная(закрытие цены). Зафиксировав окно размера (32,58), мы срезаем с матрицы A первые 32 строки и все 58 колонок $A[0:32, :]$, получая первое наблюдение для train выборки. Затем, отбираем элемент $A[33, 0]$, это будет target значение нашей целевой переменной в прогнозируемый момент времени, по которому мы будем рассчитывать Loss. На следующем шаге, мы берём $A[1:33, :]$ в качестве ещё одного наблюдения для train выборки и $A[34, 0]$ в качестве target значения для расчёта Loss-функции. Таким образом, нарезая все $N*80$ строк матрицы A , мы получаем тренировочную выборку

со следующими параметрами:

$$X_{\text{train}}.shape = (N*0.8-32+1, 32, 58)$$

$$Y_{\text{train}}.shape = (N*0.8-32+1, 32, 1)$$

Далее, перед началом обработки валидационной выборки, нам необходимо сделать отступ равный 32 строкам, после чего начать нарезать валидационные данные. Данная мера обоснована тем, что последние значения X_{train} и первые значения X_{val} будут пересекаться, а значит, произойдет утечка информации и модель будет валидироваться на части данных, которые она уже видела. Итого, после создания отступа, у нас остаётся матрица $A_{N*0.2-32, 58}$, которую нарезаем таким же окном, как и до этого. В итоге получим валидационную выборку, с размерами:

$$X_{\text{val}}.shape = (N*0.2-32-32+1, 32, 58)$$

$$Y_{\text{val}}.shape = (N*0.2-32-32+1, 32, 1)$$

Аналогичные расчёты справедливы и для испытания 1 и для испытания 3. Обратим внимание, что в данном трехмерном тензоре по первой координате мы дважды вычитаем 32, делая это в первый раз для учёта окна, во второй раз для учёта отступа, предотвращающего утечку.

Получив тренировочный и валидационный наборы данных, начинаем переходить к тестированию четырёх архитектур:

1. LSTM
2. Transformer(Encoder)
3. Transformer(Encoder+Decoder)
4. TSMixer

Для каждого значения $\lambda \in \{0, 1, 5, 10, 12, 25, 35, 45, 50\}$ мы проводим обучение моделей на трёх испытаниях и фиксируем достигнутое значение Loss-функции на каждом из них.

Так как мощность множества значений λ равно 9, количество моделей = 4, а количество испытаний = 3, то после проведения эксперимента мы получим 9 матриц $B_{3,4}$. В конце, мы проведём покоординатное усреднение всех 9 матриц, получив матрицу $B_{3,4}^*$, которая будет отражать обобщённое качество каждой модели на каждом испытании при всех исследуемых параметрах λ .

Поскольку спецификации моделей и процедуры обучения одинаковы для обеих итераций, то обсудим их в данной секции.

Parameter	Value
Batch Size	64
Learning Rate	0.0009
Scheduler	ReduceLROnPlateau
Scheduler Parameters	factor=0.5, patience=3
Number of Epochs	30
Optimizer	Adam
Loss Function	CustomLoss
Regularization	Dropout (0.15)
Validation Split	0.2

Таблица 4.1: Параметры тренировки

Таблица 4.1 описывает основные параметры процесса обучения, такие как размер батча, learning rate, количество эпох и прочие.

Parameter	Value
Input size	1 при И.1, 58 при И.2, 42 при И.3
Hidden size	1024
Num layers	3
Dropout	0.1

Таблица 4.2: Параметры LSTM модели

```
OrderedDict([('lstm', LSTM(1, 1024, num_layers=3, batch_first=True)),
             ('dropout', Dropout(p=0.1, inplace=False)),
             ('fc1', Linear(in_features=1024, out_features=32, bias=True)),
             ('fc2', Linear(in_features=32, out_features=1, bias=True))])
OrderedDict([('lstm', LSTM(58, 1024, num_layers=3, batch_first=True)),
             ('dropout', Dropout(p=0.1, inplace=False)),
             ('fc1', Linear(in_features=1024, out_features=32, bias=True)),
             ('fc2', Linear(in_features=32, out_features=1, bias=True))])
OrderedDict([('lstm', LSTM(42, 1024, num_layers=3, batch_first=True)),
             ('dropout', Dropout(p=0.1, inplace=False)),
             ('fc1', Linear(in_features=1024, out_features=32, bias=True)),
             ('fc2', Linear(in_features=32, out_features=1, bias=True))])
```

Рис. 4.1: Параметры LSTM для всех испытаний

Таблица 4.2 и рисунок 4.1 описывают архитектуру моделей LSTM, участвующих в эксперименте. Как можно видеть, модели схожи друг с другом за исключением размера входного слоя, который был адаптирован у каждой модели под специфику конкретного испытания.

Parameter	Value
Input size	1 при И.1, 58 при И.2, 42 при И.3
Model dim	512
Num heads	8
Num layers	2
Dropout	0.1

Таблица 4.3: Параметры Transformer(Encoder) модели

```

OrderedDict([('ebedder', Linear(in_features=58, out_features=512, bias=True)),
    ('pos_encoder',
        PositionalEncoding(
            (dropout): Dropout(p=0.1, inplace=False)
        )),
    ('transformer_encoder',
        TransformerEncoder(
            (layers): ModuleList(
                (0-1): 2 x TransformerEncoderLayer(
                    (self_attn): MultiheadAttention(
                        (out_proj): NonDynamicallyQuantizableLinear(in_features=512, out_features=512, bias=True)
                    )
                    (linear1): Linear(in_features=512, out_features=2048, bias=True)
                    (dropout): Dropout(p=0.1, inplace=False)
                    (linear2): Linear(in_features=2048, out_features=512, bias=True)
                    (norm1): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
                    (norm2): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
                    (dropout1): Dropout(p=0.1, inplace=False)
                    (dropout2): Dropout(p=0.1, inplace=False)
                )
            )
        )),
    ('head', Linear(in_features=512, out_features=1, bias=True))])

OrderedDict([('ebedder', Linear(in_features=42, out_features=512, bias=True)),
    ('pos_encoder',
        PositionalEncoding(
            (dropout): Dropout(p=0.1, inplace=False)
        )),
    ('transformer_encoder',
        TransformerEncoder(
            (layers): ModuleList(
                (0-1): 2 x TransformerEncoderLayer(
                    (self_attn): MultiheadAttention(
                        (out_proj): NonDynamicallyQuantizableLinear(in_features=512, out_features=512, bias=True)
                    )
                    (linear1): Linear(in_features=512, out_features=2048, bias=True)
                    (dropout): Dropout(p=0.1, inplace=False)
                    (linear2): Linear(in_features=2048, out_features=512, bias=True)
                    (norm1): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
                    (norm2): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
                    (dropout1): Dropout(p=0.1, inplace=False)
                    (dropout2): Dropout(p=0.1, inplace=False)
                )
            )
        )),
    ('head', Linear(in_features=512, out_features=1, bias=True))])

```

Рис. 4.2: Параметры Transformer(Encoder)

Таблица 4.3 и рисунок 4.2 описывают архитектуру Transformer модели с Encoder частью. Как и в прошлая модель, Transformer отличается лишь размерами входного слоя. На рисунке представлены архитектуры моделей для испытаний 2 и 3. Аналогично предыдущим моделям, для её адаптации к условиям других испытаний достаточно изменить входной слой.

Parameter	Value
Input size	1 при И.1, 58 при И.2, 42 при И.3
Model dim	512
Num heads	8
Num layers	2
Dropout	0.1

Таблица 4.4: Параметры Transformer(Encoder+Decoder) модели

```

OrderedDict([('ebedder1', Linear(in_features=58, out_features=512, bias=True)),
    ('ebedder2', Linear(in_features=58, out_features=512, bias=True)),
    ('pos_encoder',
        PositionalEncoding(
            (dropout): Dropout(p=0.1, inplace=False)
        )),
    ('transformer_encoder',
        TransformerEncoder(
            (layers): ModuleList(
                (0): TransformerEncoderLayer(
                    (self_attn): MultiheadAttention(
                        (out_proj): NonDynamicallyQuantizableLinear(in_features=512, out_features=512, bias=True)
                    )
                    (linear1): Linear(in_features=512, out_features=2048, bias=True)
                    (dropout): Dropout(p=0.1, inplace=False)
                    (linear2): Linear(in_features=2048, out_features=512, bias=True)
                    (norm1): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
                    (norm2): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
                    (dropout1): Dropout(p=0.1, inplace=False)
                    (dropout2): Dropout(p=0.1, inplace=False)
                )
            )
        )),
    ('transformer_decoder',
        TransformerDecoder(
            (layers): ModuleList(
                (0): TransformerDecoderLayer(
                    (self_attn): MultiheadAttention(
                        (out_proj): NonDynamicallyQuantizableLinear(in_features=512, out_features=512, bias=True)
                    )
                    (multihead_attn): MultiheadAttention(
                        (out_proj): NonDynamicallyQuantizableLinear(in_features=512, out_features=512, bias=True)
                    )
                    (linear1): Linear(in_features=512, out_features=2048, bias=True)
                    (dropout): Dropout(p=0.1, inplace=False)
                    (linear2): Linear(in_features=2048, out_features=512, bias=True)
                    (norm1): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
                    (norm2): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
                    (norm3): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
                    (dropout1): Dropout(p=0.1, inplace=False)
                    (dropout2): Dropout(p=0.1, inplace=False)
                    (dropout3): Dropout(p=0.1, inplace=False)
                )
            )
        )),
    ('head', Linear(in_features=512, out_features=1, bias=True)))
])

```

Рис. 4.3: Параметры Transformer(Encoder+Decoder)

Таблица 4.4 и рисунок 4.3 описывают архитектуру Transformer модели с Encoder и Decoder частями. На рисунке представлена архитектура модели для испытания 2. Аналогично предыдущим моделям, для её адаптации к условиям других испытаний достаточно изменить входной слой.

Parameter	Value
Input size	1 при И.1, 58 при И.2, 42 при И.3
Seq. lenght	32
Forecast horizon	1
Num layers	1
Dropout	0.1

Таблица 4.5: Параметры TSMixer модели

```

OrderedDict([('MixerBlock',
    ModuleList(
        (0): MixerLayer(
            (TimeMixer): TimeMixer(
                (Norm1): BatchNorm1d(1, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
                (MLPtime): Linear(in_features=32, out_features=32, bias=True)
                (Relu): ReLU()
                (dropout): Dropout(p=0.1, inplace=False)
            )
            (FeatureMixer): FeatureMixer(
                (Norm1): BatchNorm1d(1, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
                (MLPfeat1): Linear(in_features=1, out_features=1, bias=True)
                (MLPfeat2): Linear(in_features=1, out_features=1, bias=True)
                (Relu): ReLU()
                (dropout): Dropout(p=0.1, inplace=False)
            )
        )
    )),
    ('TemporalProjection',
        TemporalProjection(
            (FC): Linear(in_features=32, out_features=1, bias=True)
        )))
))

OrderedDict([('MixerBlock',
    ModuleList(
        (0): MixerLayer(
            (TimeMixer): TimeMixer(
                (Norm1): BatchNorm1d(58, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
                (MLPtime): Linear(in_features=32, out_features=32, bias=True)
                (Relu): ReLU()
                (dropout): Dropout(p=0.1, inplace=False)
            )
            (FeatureMixer): FeatureMixer(
                (Norm1): BatchNorm1d(58, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
                (MLPfeat1): Linear(in_features=58, out_features=58, bias=True)
                (MLPfeat2): Linear(in_features=58, out_features=58, bias=True)
                (Relu): ReLU()
                (dropout): Dropout(p=0.1, inplace=False)
            )
        )
    )),
    ('TemporalProjection',
        TemporalProjection(
            (FC): Linear(in_features=32, out_features=1, bias=True)
        )))
))

OrderedDict([('MixerBlock',
    ModuleList(
        (0): MixerLayer(
            (TimeMixer): TimeMixer(
                (Norm1): BatchNorm1d(42, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
                (MLPtime): Linear(in_features=32, out_features=32, bias=True)
                (Relu): ReLU()
                (dropout): Dropout(p=0.1, inplace=False)
            )
            (FeatureMixer): FeatureMixer(
                (Norm1): BatchNorm1d(42, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
                (MLPfeat1): Linear(in_features=42, out_features=42, bias=True)
                (MLPfeat2): Linear(in_features=42, out_features=42, bias=True)
                (Relu): ReLU()
                (dropout): Dropout(p=0.1, inplace=False)
            )
        )
    )),
    ('TemporalProjection',
        TemporalProjection(
            (FC): Linear(in_features=32, out_features=1, bias=True)
        )))
))

```

Рис. 4.4: Параметры TSMixer

Таблица 4.5 и рисунок 4.4 описывают архитектуру TSMixer модели. Аналогично предыдущим моделям, от испытания к испытанию модели отличаются лишь одним параметром.

4.1.1 Итерация 1

В рамках первой итерации мы работаем на данных 2020 и 2021 годов. Набор данных имеет размер 172.129 строк и (1/58/42) столбцов, где 1 столбец это цена закрытия(целевая переменная), а остальные - признаковое описание. После процесса нарезки данных окном в 32 шага, мы получаем тензор размерности (136327, 32, (1/58/42)) на тренировку и тензор размерности (33102, 32, (1/58/42)) на валидацию.

Таблица 4.6: Таблица усреднённых результатов(20-21 годы)

	И.1	И.2	И.3
LSTM	24.55	19.47	21.7
Transformer(encoder часть)	20.11	12.19	16.33
Transformer(encoder + decoder)	20.00	13.17	16.4
TSMixer	21.00	15.71	16.52

Таблица 4.6 показывает усреднённые по всем $\lambda \in \{0,1,5,10,12,25,35,45,50\}$ значения функции потерь для каждой модели и каждого испытания. Значения параметра λ выбирались исходя из эмпирических результатов, показавших эффективную границу параметра, лежащую от 0 до 50. Как можно заметить, наилучшего результата смогла достичнуть Transformer(Encoder) архитектура в испытании 2, где в качестве признаков выступают технические индикаторы. Это значит, что следующий этап(оптимизация гиперпараметров стратегии) будет проходить на её основе и с применением данного признакового пространства. Наглядно видно, как модели начинают ухудшать свой результат при отсутствии признаков. Это наблюдение подтверждает гипотезу о том, что признаки несут в себе информационную пользу для модели и помогают прогнозировать будущие цены точнее.

4.1.2 Итерация 2

В рамках второй итерации мы работаем на данных 2020, 2021 и 2022 годов. Набор данных имеет размер 241.978 строк и (1/58/42) столбцов, где 1 столбец это цена закрытия(целевая переменная), а остальные - признаковое описание. После процесса нарезки данных окном в 32 шага, мы получаем тензор размерности (189544, 32, (1/58/42)) на тренировку и тензор размерности (45487, 32, (1/58/42)) на валидацию.

Таблица 4.7: Таблица усреднённых результатов(20-21 годы)

	И.1	И.2	И.3
LSTM	28.67	22.46	24.85
Transformer(encoder часть)	24.54	17.2	18.26
Transformer(encoder + decoder)	24.4	18.55	19.00
TSMixer	25.64	18.9	20.09

Таблица 4.7 показывает усреднённые по всем $\lambda \in \{0,1,5,10,12,25,35,45,50\}$ значения функции потерь для каждой модели и каждого испытания. Как можно заметить, снова наилучшего результата смогла добиться Transformer(Encoder) архитектура в испытании 2, где в качестве признаков выступают технические индикаторы. Это значит, что следующий этап(оптимизация гиперпараметров стратегии) также будет проходить на её основе и с применением данного признакового пространства. Результаты 1 этапа данной итерации подтверждают результаты этого же этапа предыдущей итерации 4.6, где модели продемонстрировали схожее поведение. Во время второй итерации видно, что каждая из моделей ухудшила свой средний loss на каждом из испытаний. Причиной этому может служить появление данных 2022 года. Данные 2022 года были обрезаны по 1 марта и причиной данного решения является тот факт,

что рынок того времени не отражал типичное поведение игроков. На тот момент на рынке преобладал несистематический риск, а, следовательно, данные того периода не содержат в себе полезной информации для модели. Однако стоит заметить, что несмотря на обрезанные по началу апреля данные, фондовый рынок имел высокую волатильность и дальше. Это могло послужить причиной повышенного значения усреднённой функции потерь.

4.2 Результаты поиска оптимальных гиперпараметров

Итого, по завершению первого этапа в обеих итерациях, мы смогли определить наиболее качественную модель с точки зрения усреднённой функции потерь, а также подобрать то признаковое описание, которое позволяет нам добиться наименьшего значения функции потерь для данной модели. В ходе проведения испытаний мы выяснили, что в обеих итерациях модель Transformer(Encoder) показала наилучший результат на испытании 2, где признаками выступают 57 технических индикаторов. Следовательно, следующей нашей задачей будет являться интеграция модели в торговый алгоритм и оптимизация последнего по гиперпараметрам λ , risk aversion(далее RA).

Имея 9 весов Трансформера, обученного на $\lambda \in \{0,1,5,10,12,25,35,45,50\}$ мы переходим к проведению бектеста и поиску оптимальных параметров на обрезанных по март данных 2022 года для первой итерации и данных 2023 года для второй итерации. Наборы данных соответствуют схеме рисунка 3.7. Используемый алгоритм описан на схеме алгоритма 1. Также стоит напомнить, что параметр RA определяется как γ в теле целевой функции, при решении задачи оптимизации 3.4 и отвечает за чувствительность к риску при поиске оптимальных торговых операций.

Обобщённая схема проведения бектестов для всех λ, γ :

Algorithm 2 Бектесты для λ, γ

Require:

$$\Lambda = \{0,1,5,10,12,25,35,45,50\}, \Gamma = \{0.5, 1, 2, 3, 4, 5, 8\};$$

Ensure:

Sharpe Ratio - значение коэффициента Шарпа для получившегося портфеля

- 1: **for** $\lambda \in \Lambda, \gamma \in \Gamma$ **do**
 - 2: Проводим почасовое прогнозирование всего года трансформером, обученным на λ .
 - 3: На полученных данных проводим бектест с $RA = \gamma$
 - 4: Фиксируем полученный Sharpe Ratio
 - 5: **end for**=0
-

Схема алгоритма 2 показывает общий вид проведения бектеста с фиксированными параметрами.

4.2.1 Итерация 1

В рамках первой итерации, сбор результатов бектеста и поиск оптимальных параметров λ, γ производится на данных 2022 года, обрезанных по 1 марта. Общий размер данных составляет 69849 строк. Транзакционные издержки равны 0,02%, что является наиболее выгодным предложением для частного трейдера на момент проведения исследования.

Lambda=0,RA=0.5,Sharpe=-2.205	Lambda=12,RA=3,Sharpe=-1.674
Lambda=0,RA=1,Sharpe=-2.217	Lambda=12,RA=4,Sharpe=-1.713
Lambda=0,RA=2,Sharpe=-2.207	Lambda=12,RA=5,Sharpe=-1.731
Lambda=0,RA=3,Sharpe=-2.218	Lambda=12,RA=8,Sharpe=-1.773
Lambda=0,RA=4,Sharpe=-2.276	Lambda=25,RA=0.5,Sharpe=-2.575
Lambda=0,RA=5,Sharpe=-2.326	Lambda=25,RA=1,Sharpe=-2.565
Lambda=0,RA=8,Sharpe=-2.408	Lambda=25,RA=2,Sharpe=-2.577
Lambda=1,RA=0.5,Sharpe=-2.484	Lambda=25,RA=3,Sharpe=-2.567
Lambda=1,RA=1,Sharpe=-2.443	Lambda=25,RA=4,Sharpe=-2.565
Lambda=1,RA=2,Sharpe=-2.432	Lambda=25,RA=5,Sharpe=-2.553
Lambda=1,RA=3,Sharpe=-2.336	Lambda=25,RA=8,Sharpe=-2.538
Lambda=1,RA=4,Sharpe=-2.26	Lambda=35,RA=0.5,Sharpe=-0.908
Lambda=1,RA=5,Sharpe=-2.201	Lambda=35,RA=1,Sharpe=-0.9
Lambda=1,RA=8,Sharpe=-2.104	Lambda=35,RA=2,Sharpe=-0.828
Lambda=5,RA=0.5,Sharpe=-2.647	Lambda=35,RA=3,Sharpe=-0.825
Lambda=5,RA=1,Sharpe=-2.585	Lambda=35,RA=4,Sharpe=-0.865
Lambda=5,RA=2,Sharpe=-2.529	Lambda=35,RA=5,Sharpe=-0.903
Lambda=5,RA=3,Sharpe=-2.491	Lambda=35,RA=8,Sharpe=-0.984
Lambda=5,RA=4,Sharpe=-2.472	Lambda=45,RA=0.5,Sharpe=-1.901
Lambda=5,RA=5,Sharpe=-2.496	Lambda=45,RA=1,Sharpe=-1.893
Lambda=5,RA=8,Sharpe=-2.427	Lambda=45,RA=2,Sharpe=-1.878
Lambda=10,RA=0.5,Sharpe=-1.128	Lambda=45,RA=3,Sharpe=-1.878
Lambda=10,RA=1,Sharpe=-1.098	Lambda=45,RA=4,Sharpe=-1.888
Lambda=10,RA=2,Sharpe=-1.054	Lambda=45,RA=5,Sharpe=-1.868
Lambda=10,RA=3,Sharpe=-1.042	Lambda=45,RA=8,Sharpe=-1.802
Lambda=10,RA=4,Sharpe=-1.098	Lambda=50,RA=0.5,Sharpe=-3.322
Lambda=10,RA=5,Sharpe=-1.212	Lambda=50,RA=1,Sharpe=-3.354
Lambda=10,RA=8,Sharpe=-1.478	Lambda=50,RA=2,Sharpe=-3.35
Lambda=12,RA=0.5,Sharpe=-1.614	Lambda=50,RA=3,Sharpe=-3.347
Lambda=12,RA=1,Sharpe=-1.621	Lambda=50,RA=4,Sharpe=-3.34
Lambda=12,RA=2,Sharpe=-1.62	Lambda=50,RA=5,Sharpe=-3.385
Lambda=12,RA=3,Sharpe=-1.674	Lambda=50,RA=8,Sharpe=-3.387

Рис. 4.5: Результаты серии бектестов для первой итерации

Как видно из рисунка 4.5, ни на одной из комбинаций гиперпараметров алгоритм не смог достичь положительной доходности. Как говорилось ранее, в течение всего 2022 года российский фондовый рынок страдал от высокой волатильности вследствие сильного несистематического риска, вызванного санкционным давлением на российскую экономику. Данные факторы могли внести негативный вклад в успех торгового алгоритма, однако, нашей задачей по прежнему является поиск параметров, дающих максимальный коэффициент Шарпа. Как видно на рисунке 4.5, оптимальными параметрами являются $\lambda^* = 35$, $RA^* = 3$, именно на их мы будем тестировать торговую стратегию на следующем шаге.

4.2.2 Итерация 2

В рамках второй итерации, сбор результатов бектеста и поиск оптимальных параметров λ, γ производится на данных 2023 года. Общий размер данных составляет 77043 строк. Транзакционные издержки также равны 0,02%.

Lambda=0, RA=0.5, Sharpe=-0.736	Lambda=12, RA=4, Sharpe=-2.456
Lambda=0, RA=1, Sharpe=-0.786	Lambda=12, RA=5, Sharpe=-2.466
Lambda=0, RA=2, Sharpe=-0.798	Lambda=12, RA=8, Sharpe=-2.502
Lambda=0, RA=3, Sharpe=-0.784	Lambda=25, RA=0.5, Sharpe=-1.821
Lambda=0, RA=4, Sharpe=-0.822	Lambda=25, RA=1, Sharpe=-1.798
Lambda=0, RA=5, Sharpe=-0.873	Lambda=25, RA=2, Sharpe=-1.785
Lambda=0, RA=8, Sharpe=-1.004	Lambda=25, RA=3, Sharpe=-1.791
Lambda=1, RA=0.5, Sharpe=-1.01	Lambda=25, RA=4, Sharpe=-1.814
Lambda=1, RA=1, Sharpe=-0.985	Lambda=25, RA=5, Sharpe=-1.833
Lambda=1, RA=2, Sharpe=-0.918	Lambda=25, RA=8, Sharpe=-1.868
Lambda=1, RA=3, Sharpe=-0.885	Lambda=35, RA=0.5, Sharpe=-1.819
Lambda=1, RA=4, Sharpe=-0.9	Lambda=35, RA=1, Sharpe=-1.827
Lambda=1, RA=5, Sharpe=-0.894	Lambda=35, RA=2, Sharpe=-1.861
Lambda=1, RA=8, Sharpe=-0.869	Lambda=35, RA=3, Sharpe=-1.9
Lambda=5, RA=0.5, Sharpe=0.215	Lambda=35, RA=4, Sharpe=-1.923
Lambda=5, RA=1, Sharpe=0.242	Lambda=35, RA=5, Sharpe=-1.947
Lambda=5, RA=2, Sharpe=0.271	Lambda=35, RA=8, Sharpe=-1.894
Lambda=5, RA=3, Sharpe=0.276	Lambda=45, RA=0.5, Sharpe=-1.221
Lambda=5, RA=4, Sharpe=0.275	Lambda=45, RA=1, Sharpe=-1.224
Lambda=5, RA=5, Sharpe=0.345	Lambda=45, RA=2, Sharpe=-1.254
Lambda=5, RA=8, Sharpe=0.334	Lambda=45, RA=3, Sharpe=-1.279
Lambda=10, RA=0.5, Sharpe=-2.291	Lambda=45, RA=4, Sharpe=-1.283
Lambda=10, RA=1, Sharpe=-2.285	Lambda=45, RA=5, Sharpe=-1.279
Lambda=10, RA=2, Sharpe=-2.245	Lambda=45, RA=8, Sharpe=-1.265
Lambda=10, RA=3, Sharpe=-2.209	Lambda=50, RA=0.5, Sharpe=-1.067
Lambda=10, RA=4, Sharpe=-2.189	Lambda=50, RA=1, Sharpe=-1.063
Lambda=10, RA=5, Sharpe=-2.16	Lambda=50, RA=2, Sharpe=-1.081
Lambda=10, RA=8, Sharpe=-2.21	Lambda=50, RA=3, Sharpe=-1.1
Lambda=12, RA=0.5, Sharpe=-2.435	Lambda=50, RA=4, Sharpe=-1.144
Lambda=12, RA=1, Sharpe=-2.438	Lambda=50, RA=5, Sharpe=-1.168
Lambda=12, RA=2, Sharpe=-2.443	Lambda=50, RA=8, Sharpe=-1.246
Lambda=12, RA=3, Sharpe=-2.445	

Рис. 4.6: Результаты серии бектестов для второй итерации

На рисунке 4.6 можно наблюдать, что наибольший коэффициент Шарпа достигается при $\lambda^* = 5$, $RA^* = 5$. Именно на них будет проводиться тестирование торговой стратегии. Следует заметить, что по сравнению со первой итерацией, оптимальные λ и RA поменялись, а значит, мы можем констатировать факт того, что стратегия адаптируется в зависимости от рынка, на котором происходит торговля. Нестабильный рынок 2022 года требовал увеличивать параметр λ из-за высокой нестабильности и непредсказуемости рынка, в то время как растущий рынок 2023 года позволил снизить λ , больше концентрируясь на точность прогноза, а не доле правильно угаданных направлений.

4.3 Результаты тестирования торговой стратегии

Наконец, последним этапом проверки торговой стратегии является её тестирование на тестовых данных, которых она не видела прежде. Дойдя до этого этапа, мы уже выбрали оптимальную нейросетевую архитектуру, а также оптимальные гиперпараметры торговой стратегии λ^* и RA^* . Зафиксировав архитектуру и гиперпараметры, мы проводим бектест на тестовых данных, в ходе которого собираем результаты торговли.

Ключевой метрикой оценки качества стратегии является Return - доходность стратегии за тестируемый период. Кроме того, результат стратегии сравнивается с результатом доходности бенчмарка (Индекс Мосбиржи) для ответа на вопрос о том, может ли она перегнать бенчмарк в виде индекса, торгуя лишь активами из него самого.

4.3.1 Итерация 1

Тестирование стратегии с параметрами $\lambda^* = 35$, $RA^* = 3$ в рамках первой итерации производится на данных 2023 года, имеющих 77043 наблюдения. Транзакционные издержки по прежнему равны 0,02%.

Metric	Value
Initial value (cash)	1.000e+06
Final value (cash)	9.585e+05
Profit (cash)	-4.151e+04
Avg. return (annualized)	-0.1%
Volatility (annualized)	32.1%
Avg. excess return (annualized)	-0.1%
Avg. active return (annualized)	-0.1%
Excess volatility (annualized)	32.1%
Active volatility (annualized)	32.1%
Avg. growth rate (annualized)	-5.2%
Avg. excess growth rate (annualized)	-5.2%
Avg. active growth rate (annualized)	-5.2%
Sharpe ratio	-0.00
Information ratio	-0.00
Avg. drawdown	-9.2%
Min. drawdown	-24.9%
Avg. leverage	101.4%
Max. leverage	106.8%
Avg. turnover	95.9%
Max. turnover	104.1%

Таблица 4.8: Результаты тестирования для $\lambda = 35$, $RA = 3$ ($c=0.0002$)

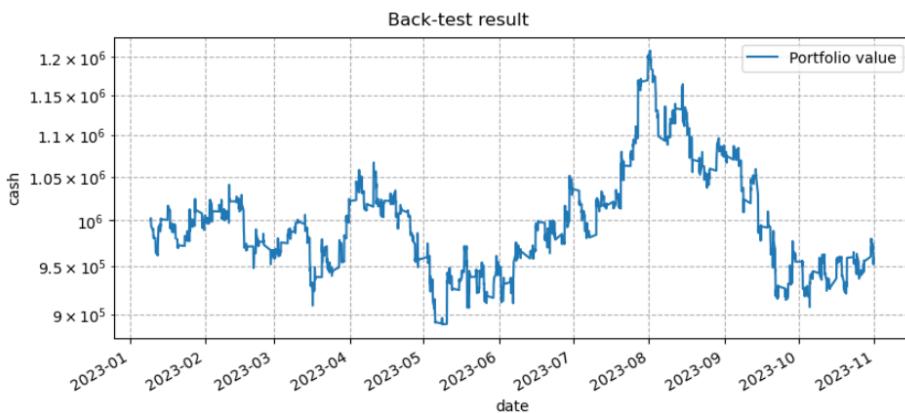


Рис. 4.7: График стоимости портфеля $\lambda = 35$, $RA = 3$ ($c=0.0002$)

Рисунки 4.8 и 4.7 содержат результаты тестирования стратегии. Мы можем наблюдать, что алгоритм не смог реализовать положительную доходность, хотя период с 2023-06 по 2023-08 можно охарактеризовать периодом уверенного роста цены портфеля. Разбираясь в причинах такого неудовлетворительного результата, следует обратить внимание на общую стоимость транзакционных издержек, которые мы заплатили в ходе торговли: 712796 рублей. Как можно увидеть, на комиссию брокеру

уходит достаточно большая доля, а значит, что алгоритм, чрезмерно часто перебалансируя портфель, теряет на этом большие суммы.

Проведём эксперимент и понизим комиссию с 0,02% до 0,01%:

Metric	Value
Initial value (cash)	1.000e+06
Final value (cash)	1.358e+06
Profit (cash)	3.585e+05
Avg. return (annualized)	43.1%
Volatility (annualized)	32.1%
Avg. excess return (annualized)	43.1%
Avg. active return (annualized)	43.1%
Excess volatility (annualized)	32.1%
Active volatility (annualized)	32.1%
Avg. growth rate (annualized)	37.9%
Avg. excess growth rate (annualized)	37.9%
Avg. active growth rate (annualized)	37.9%
Sharpe ratio	1.34
Information ratio	1.34
Avg. drawdown	-6.1%
Min. drawdown	-19.1%
Avg. leverage	101.3%
Max. leverage	106.3%
Avg. turnover	96.1%
Max. turnover	103.9%

Таблица 4.9: Результаты тестирования для $\lambda = 35$, RA = 3 (c=0.0001)

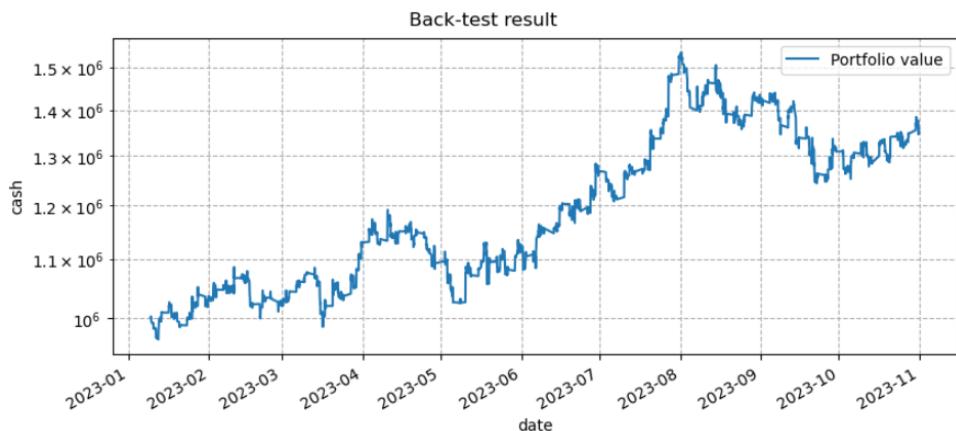


Рис. 4.8: График стоимости портфеля $\lambda = 35$, RA = 3 (c=0.0001)

Понизив комиссию брокера до 0,01% мы видим, как алгоритм начинает приносить доходность, о чем свидетельствуют рисунки 4.9 и 4.8. Общая комиссия брокеру, в свою очередь, падает до 428552 рублей. За аналогичный период, бенчмарк (Индекс Мосбиржи) принёс доходность в 44.65%, в то время как этот алгоритм принес доходность в 35.8%

Понизим издержки с 0,01% до 0,009%:

Metric	Value
Initial value (cash)	1.000e+06
Final value (cash)	1.459e+06
Profit (cash)	4.594e+05
Avg. return (annualized)	51.9%
Volatility (annualized)	32.1%
Avg. excess return (annualized)	51.9%
Avg. active return (annualized)	51.9%
Excess volatility (annualized)	32.1%
Active volatility (annualized)	32.1%
Avg. growth rate (annualized)	46.8%
Avg. excess growth rate (annualized)	46.8%
Avg. active growth rate (annualized)	46.8%
Sharpe ratio	1.62
Information ratio	1.62
Avg. drawdown	-5.5%
Min. drawdown	-18.0%
Avg. leverage	101.2%
Max. leverage	105.7%
Avg. turnover	96.1%
Max. turnover	103.3%

Таблица 4.10: Результаты тестирования для $\lambda = 35$, RA = 3 (c=0.00009)

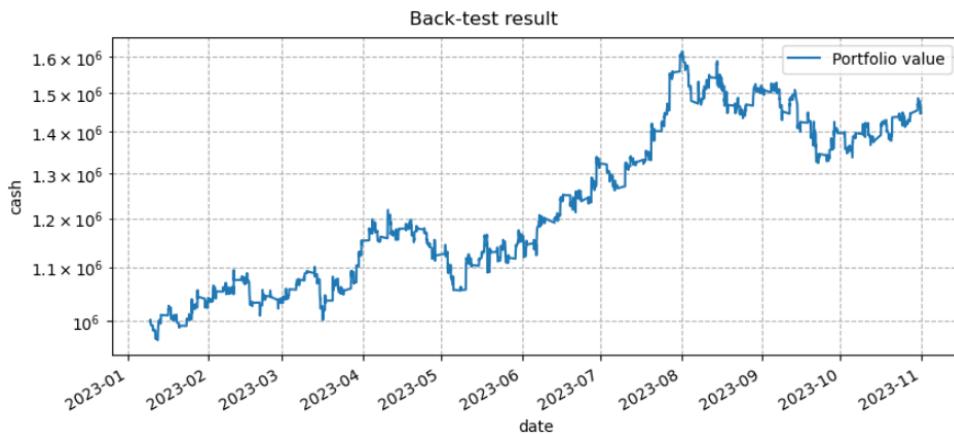


Рис. 4.9: График стоимости портфеля $\lambda = 35$, RA = 3 (c=0.00009)

Доходность алгоритма составила 45.9%, что превышает доходность бенчмарка(44.65%), а общая комиссия составила 356463 рублей. Оценивая полученный результат можно сказать, что слишком высокая комиссия мешает алгоритму вести прибыльную торговлю.

4.3.2 Итерация 2

Тестирование стратегии с параметрами $\lambda^* = 5$, RA* = 5 в рамках второй итерации производится на данных 2024 года (01.01.2024 - 01.04.2024), имеющих 20803 наблюдения. Транзакционные издержки по прежнему равны 0,02%.

Metric	Value
Initial value (cash)	1.000e+06
Final value (cash)	1.081e+06
Profit (cash)	8.062e+04
Avg. return (annualized)	39.5%
Volatility (annualized)	28.7%
Avg. excess return (annualized)	39.5%
Avg. active return (annualized)	39.5%
Excess volatility (annualized)	28.7%
Active volatility (annualized)	28.7%
Avg. growth rate (annualized)	35.4%
Avg. excess growth rate (annualized)	35.4%
Avg. active growth rate (annualized)	35.4%
Sharpe ratio	1.37
Information ratio	1.37
Avg. drawdown	-4.1%
Min. drawdown	-12.3%
Avg. leverage	100.5%
Max. leverage	108.5%
Avg. turnover	87.2%
Max. turnover	104.2%

Таблица 4.11: Результаты тестирования для $\lambda = 5$, RA = 5 (c=0.0002)

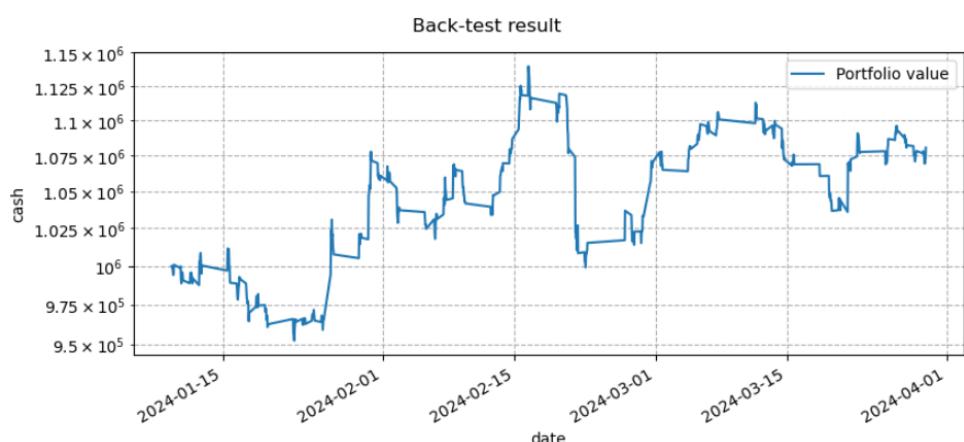


Рис. 4.10: График стоимости портфеля $\lambda = 5$, RA = 5 (c=0.0002)

Результаты рисунков 4.11 и 4.10 говорят об эффективности торгового алгоритма при транзакционных издержках в 0,02%. Начиная с января 2024 и по начало апреля этого же года, стоимость портфеля выросла на 8.1%. За аналогичный период времени, бенчмарк (Индекс Мосбиржи) вырос на 5,6%, что свидетельствует об эффективности алгоритма. Общие издержки на комиссию составили 88629 рублей.

Не смотря на зафиксированную положительную доходность, проверим алгоритм на более низких комиссиях. Для начала, понизим комиссию до 0,01%:

Metric	Value
Initial value (cash)	1.000e+06
Final value (cash)	1.083e+06
Profit (cash)	8.321e+04
Avg. return (annualized)	40.6%
Volatility (annualized)	28.7%
Avg. excess return (annualized)	40.6%
Avg. active return (annualized)	40.6%
Excess volatility (annualized)	28.7%
Active volatility (annualized)	28.7%
Avg. growth rate (annualized)	36.5%
Avg. excess growth rate (annualized)	36.5%
Avg. active growth rate (annualized)	36.5%
Sharpe ratio	1.41
Information ratio	1.41
Avg. drawdown	-4.1%
Min. drawdown	-12.4%
Avg. leverage	100.5%
Max. leverage	108.4%
Avg. turnover	87.2%
Max. turnover	104.2%

Таблица 4.12: Результаты тестирования для $\lambda = 5$, RA = 5 (c=0.0001)

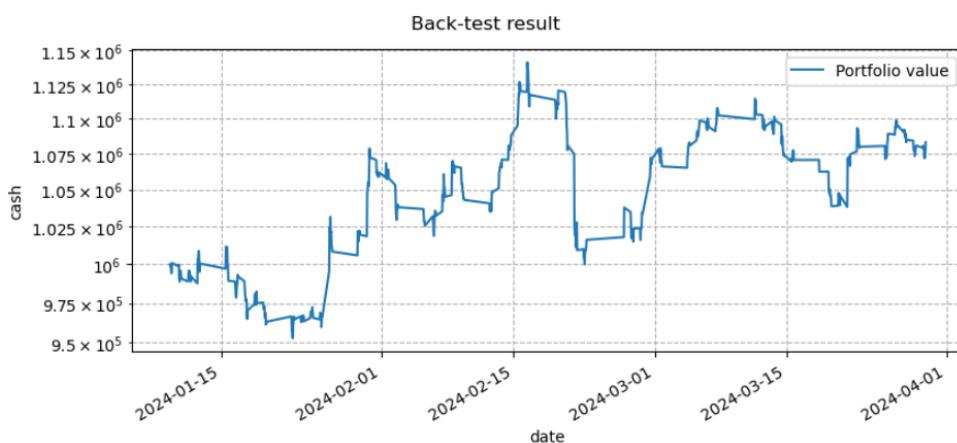


Рис. 4.11: График стоимости портфеля $\lambda = 5$, RA = 5 (c=0.0001)

На рисунках 4.12 и 4.11 мы можем наблюдать улучшенные характеристики торговли. Так, наш портфель вырос на 8.3% против 5,6% индекса. Общий размер комиссии составил 83295 рублей.

Понизим комиссию до 0,009%:

Metric	Value
Initial value (cash)	1.000e+06
Final value (cash)	1.092e+06
Profit (cash)	9.164e+04
Avg. return (annualized)	44.1%
Volatility (annualized)	28.7%
Avg. excess return (annualized)	44.1%
Avg. active return (annualized)	44.1%
Excess volatility (annualized)	28.7%
Active volatility (annualized)	28.7%
Avg. growth rate (annualized)	40.0%
Avg. excess growth rate (annualized)	40.0%
Avg. active growth rate (annualized)	40.0%
Sharpe ratio	1.54
Information ratio	1.54
Avg. drawdown	-3.9%
Min. drawdown	-12.4%
Avg. leverage	100.5%
Max. leverage	108.9%
Avg. turnover	87.2%
Max. turnover	104.2%

Таблица 4.13: Результаты тестирования для $\lambda = 5$, RA = 5 (c=0.00009)

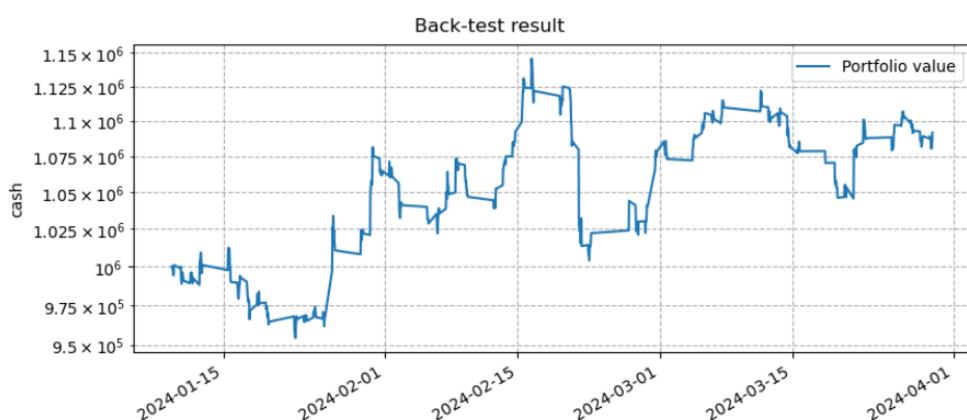


Рис. 4.12: График стоимости портфеля $\lambda = 5$, RA = 5 (c=0.00009)

При снижении комиссии до 0,009%, доходность растёт до 9,2%, а комиссия падает до 79297 рублей.

4.4 Обсуждение результатов

После проведения эксперимента и фиксации полученных результатов, мы можем переходить к их интерпретации и обсуждению.

4.4.1 Обсуждение результатов поиска оптимальной архитектуры

Приведём основные выводы данного этапа:

1. Оценивая исследуемые архитектуры по результирующим таблицам первой и второй итерации 4.6 и 4.7, мы можем заметить, что наихудшую производительность показывает архитектура LSTM, дающая наивысший средний Loss среди всех испытуемых моделей. Причиной такого результата может служить ограничение модели на объём удерживаемой памяти внутри себя, что является известной проблемой всех рекуррентных нейросетевых архитектур [51].
2. Говоря об TSMixer архитектуре, следует отметить её высокую скорость обучения по причине того, что она состоит из полносвязных слоёв, однако данный факт сыграл негативную роль в результатах модели, из-за чего её точность не смогла сравняться с точностью трансформеров, хотя и обогнала LSTM.
3. Трансформеры (оба варианта) смогли показать наилучший результат среди всех исследуемых моделей, благодаря блокам памяти (Attention Layer) и позиционному кодированию (Positional Encoding). Данные нововведения позволяют захватывать более сложные взаимосвязи во входящих последовательностях, а значит – совершать более качественные прогнозы.
4. Говоря о признаковом описании ряда, мы можем наблюдать наихудший результат у всех четырёх моделей, работающих с одномерным рядом (ряд цен без дополнительных признаков). Полученные результаты вполне очевидны, так как используя данные одних лишь цен, трудно обнаружить сложную структуру во входящей последовательности. На втором месте по результативности идёт постановка, в которой признаками являются цены закрытий других активов. Данное описание целевой переменной позволило понизить средний Loss у всех моделей. Исходя из этого, можно выдвинуть гипотезу о том, что модель использовала значения цен сильно коррелированных активов для предсказания поведения цен. Сама по себе корреляция активов не является гипотезой и давно исследуется научным сообществом [52]. Наконец, наилучший результат с точки зрения точности на обеих итерациях дали 57 технических индикаторов, которые были получены вместе с ценой через API Мосбиржи. Данные индикаторы позволили всем моделям добиться оптимальных результатов в прогнозировании, а значит, предоставили скрытую в них информацию о поведении цены.

4.4.2 Обсуждение результатов поиска оптимальных гиперпараметров торговой стратегии

Обсуждая результаты поиска оптимальных гиперпараметров, следует обратить внимание на то, что они отличаются от итерации к итерии, а значит адаптируются под конъюнктуру рынка и данных, на которых обучается модель. Попробуем проинтерпретировать полученные результаты.

1. Первая итерация показала, что оптимальными параметрами торговли на рынке 2022 года оказались $\lambda^* = 35$, $RA^* = 3$. Напоминаем, что в рамках эксперимента, $\lambda \in \{0, 1, 5, 10, 12, 25, 35, 45, 50\}$, $RA \in \{0.5, 1, 2, 3, 4, 5, 8\}$, а значит, полученная λ^* имеет достаточно высокое значение, а RA^* имеет среднее значение. Можно предположить, что данная необходимость обоснована высокой волатильностью и

непредсказуемостью рынка 2022 года, когда санкционное давление на экономику России вызвало высокий несистематический риск на всех её рынках. Этот факт заставляет нашу модель обращать особое внимание на направление движения предсказанного актива (большой λ^*), так как цены ведут себя нехарактерным для спокойного времени образом. В свою очередь, средний RA^* сигнализирует нам о необходимости учёта рисков со средней силой. Иными словами, алгоритм должен вести торговлю с умеренной агрессией, хотя и не совсем консервативно.

2. Вторая итерация показала, что оптимальными параметрами торговли на рынке 2023 года оказались $\lambda^* = 5$, $RA^* = 5$. Можно сказать, что в условиях растущего рынка с положительной динамикой, модель начинает меньше опасаться непредсказуемого поведения цен, проявляя это в снижении параметра λ^* . Однако, параметр RA^* возрос по сравнению со первой итерацией, что означает, что теперь требуется вести более менее рискованную торговлю.

Можно заключить, что оптимальные параметры в, большинстве своём, неплохо соотносятся с реальностью.

4.4.3 Обсуждение результатов тестирования стратегии

Результаты тестирования стратегий сообщают об одной основной уязвимости нашего алгоритма, которая в некоторых случаях мешает ему вести прибыльную торговлю – высокие издержки на комиссию брокеру. Обсудим это подробнее:

1. Согласно результатам, наблюдаемым на 4.8, 4.7, 4.9, 4.8 алгоритм первой итерации торгует либо в минус, либо не догоняет бенчмарк, по причине высоких издержек на комиссию и лишь с комиссии, равной 0.009%, наш алгоритм начинает обгонять бенчмарк на тестовых данных. Данные издержки происходят из-за частых перебалансировок портфеля, которые приводят к большому количеству сделок на рынке, не все из которых могут покрыть комиссию. Решением данной проблемы может служить переход на данных большей фракции, например на двухчасовые или дневные диапазоны.
2. Более того, гипотетической причиной неудачи на первой итерации может служить то, что оптимальный параметр Loss функции, а также коэффициент RA , подбирались на рынке 2022 года, который кардинально отличался от рынка 2023 года, на котором эти параметры тестировались. Данное различие двух рынков, предположительно, могло спровоцировать негативный результат при тестировании. Частично, эта гипотеза подтверждается результатами второй итерации, в которой наша стратегия, оптимизированная на растущем рынке 2023 года, смогла продемонстрировать положительный результат на части 2024 года, обогнав бенчмарк примерно в два раза (индекс +5.6%, стратегия +8.1%).

4.4.4 Результаты работы и перспективы её развития

В ходе проделанной работы, было произведено:

1. Адаптация различных нейросетевых архитектур под задачу прогнозирования временного ряда
2. Разработка функции потерь для их обучения
3. Создание торговой стратегии на основе исследуемых моделей

-
4. Оценка эффективности прогнозов исследуемых моделей
 5. Оценка эффективности торговой стратегии

Исходя из проделанной работы можно заключить, что архитектуры типа Transformer, обученные на разработанной функции потерь, способны производить прогнозы, достаточно точные для ведения прибыльной торговли. В свою очередь, разработанный торговый алгоритм позволяет обрабатывать прогнозы и производить перебалансировку портфеля с достаточной скоростью и точностью.

Однако, как было сказано ранее, существенным недостатком данного подхода являются большие расходы на комиссии брокеру, которые мешают ведению прибыльной торговли в некоторых случаях. Причиной таких расходов является частая перебалансировка портфеля (раз в час). Одним из возможных способов устранения данных недостатков является переход на данные с более низкой частотой обновления.

Итого, данная работа представляет из себя широкий спектр исследований, включающий в себя работу с различными математическими моделями, оптимационными задачами и исследования в области глубокого обучения. Данная работа имеет перспективы дальнейшего развития и улучшения уже имеющихся решений, так как автором была подготовлена достаточно качественная теоретическая и практическая база, основанная на современных методах анализа, обработки и прогнозирования данных.

Литература

- [1] Sezer Omer Berat, Gudelek Mehmet Ugur, Ozbayoglu Ahmet Murat. Financial Time Series Forecasting with Deep Learning : A Systematic Literature Review: 2005-2019. — 2019. — 1911.13288.
- [2] Svoboda Radek, Basterrech Sebastián, Kozal Jdrzej et al. A Natural Gas Consumption Forecasting System for Continual Learning Scenarios based on Hoeffding Trees with Change Point Detection Mechanism. — 2023. — 09.
- [3] Cheng Yuzhong, Nguyen Linh Thi Hoai, Ozaki Akinori, Ta Ton Viet. Deep learning-based method for weather forecasting: A case study in Itoshima. — 2024. — 2403.14918.
- [4] Morid Mohammad Amin, Sheng Olivia R. Liu, Dunbar Joseph. Time Series Prediction using Deep Learning Methods in Healthcare. — 2022. — 2108.13461.
- [5] Slimani Nadia, Amghar Mustapha, Sbiti N. Deep learning and time series analysis application on traffic flow forecasting // Journal of Theoretical and Applied Information Technology. — 2022. — 03. — Vol. 100. — P. 1247.
- [6] Zinenko Anna. Forecasting financial time series using singular spectrum analysis // Business Informatics. — 2023.
- [7] Staudemeyer Ralf C., Morris Eric Rothstein. Understanding LSTM – a tutorial into Long Short-Term Memory Recurrent Neural Networks. — 2019. — 1909.09586.
- [8] Chung Junyoung, Gulcehre Caglar, Cho KyungHyun, Bengio Yoshua. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. — 2014. — 1412.3555.
- [9] Wen Qingsong, Zhou Tian, Zhang Chaoli et al. Transformers in Time Series: A Survey. — 2023. — 2202.07125.
- [10] Vaswani Ashish, Shazeer Noam, Parmar Niki et al. Attention Is All You Need. — 2023. — 1706.03762.
- [11] Zeng Zhen, Kaur Rachneet, Siddagangappa Suchetha et al. Financial Time Series Forecasting using CNN and Transformer. — 2023. — 2304.04912.
- [12] Machine learning vs deep learning in stock market investment: an international evidence / Jing Hao, Feng He, Feng Ma et al. // Annals of Operations Research. — 2023. — 03. — P. 1–23.
- [13] Chen Si-An, Li Chun-Liang, Yoder Nate et al. TSMixer: An All-MLP Architecture for Time Series Forecasting. — 2023. — 2303.06053.

-
- [14] Hochreiter Sepp, Schmidhuber Jürgen. Long short-term memory // Neural computation. — 1997. — Vol. 9, no. 8. — P. 1735–1780.
 - [15] moexalgo. MoexAlgo. — <https://github.com/moexalgo/moexalgo>. — 2023.
 - [16] Bubeck Sébastien, Chandrasekaran Varun, Eldan Ronen et al. Sparks of Artificial General Intelligence: Early experiments with GPT-4. — 2023. — 2303.12712.
 - [17] SuperGLUE. GLUE Benchmark leaderboard. — <https://gluebenchmark.com/leaderboard>. — 2024. — Accessed: March 13, 2024.
 - [18] SQuAD. SQuAD Benchmark leaderboard. — <https://rajpurkar.github.io/SQuAD-explorer/>. — 2024. — Accessed: March 13, 2024.
 - [19] ImageNet. ImageNet Benchmark leaderboard. — <https://paperswithcode.com/sota/image-classification-on-imagenet>. — 2024. — Accessed: March 13, 2024.
 - [20] Garza Azul, Mergenthaler-Canseco Max. TimeGPT-1. — 2023. — 2310.03589.
 - [21] Tsay Ruey S. Analysis of Financial Time Series. — Wiley, 2010.
 - [22] Aas Kjersti, Dimakos Xeni K. Statistical Modelling of Financial Time Series: An Introduction. — Norwegian Academy of Science and Letters, 2004.
 - [23] Forecasting stock price directional movements using technical indicators: Investigating window size effects on one-step-ahead forecasting / Yauheniya Shynkevich, T.M. McGinnity, Sonya Coleman et al. — 2014. — 03.
 - [24] CAST: Using neural networks to improve trading systems based on technical analysis by means of the RSI financial indicator / Alejandro Rodríguez González, Ángel García-Crespo, Ricardo Colomo Palacios et al. // Expert Syst. Appl. — 2011. — Vol. 38. — P. 11489–11500. — URL: <https://api.semanticscholar.org/CorpusID:5956973>.
 - [25] Chavarnakul Thira, Enke David. Intelligent technical analysis based equivolume charting for stock trading using neural networks // Expert Systems with Applications. — 2008. — 02. — Vol. 34. — P. 1004–1017.
 - [26] Chenoweth Tim, Obradovic Zoran, Lee Sauchi. Embedding Technical Analysis Into Neural Network Based Trading Systems. // Applied Artificial Intelligence. — 1996. — 12. — Vol. 10. — P. 523–542.
 - [27] Lee Tae-Hwy. Loss Functions in Time series forecasting. — 2007. — 04.
 - [28] September Marcus A. K., Passino Francesco Sanna, Goldmann Leonie, Hinel Anton. Extended Deep Adaptive Input Normalization for Preprocessing Time Series Data for Neural Networks. — 2024. — 2310.14720.
 - [29] Godfrey Luke B., Gashler Michael S. Neural Decomposition of Time-Series Data for Effective Generalization // CoRR. — 2017. — Vol. abs/1705.09137. — arXiv : 1705.09137.
 - [30] Du Wenjie, Côté David, Liu Yan. SAITS: Self-attention-based imputation for time series // Expert Systems with Applications. — 2023. — . — Vol. 219. — P. 119619. — URL: <http://dx.doi.org/10.1016/j.eswa.2023.119619>.

-
- [31] Lopez de Prado Marcos. *Advances in Financial Machine Learning*. — Wiley, 2018.
 - [32] Daniel Fabrice. *Financial Time Series Data Processing for Machine Learning*. — 2019. — 1907.03010.
 - [33] Patarwal Preeti, Bala Rajni, Singh Ram. *Financial and Non-Stationary Time Series Forecasting using LSTM Recurrent Neural Network for Short and Long Horizon*. — 2019. — 07. — P. 1–7.
 - [34] Siami-Namini Sima, Namin Akbar Siami. *Forecasting Economics and Financial Time Series: ARIMA vs. LSTM*. — 2018. — 1803.06386.
 - [35] Box George EP, Jenkins Gwilym M, Reinsel Gregory C. *Time Series Analysis: Forecasting and Control // Holden-Day*. — 1970.
 - [36] Gradient-based learning applied to document recognition / Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner // Proceedings of the IEEE. — 1998. — Vol. 86, no. 11. — P. 2278–2324.
 - [37] Krizhevsky Alex, Sutskever Ilya, Hinton Geoffrey E. *ImageNet Classification with Deep Convolutional Neural Networks // Advances in Neural Information Processing Systems*. — 2012. — Vol. 25. — P. 1097–1105.
 - [38] Wu Neo, Green Bradley, Ben Xue, O’Banion Shawn. *Deep Transformer Models for Time Series Forecasting: The Influenza Prevalence Case*. — 2020. — 2001.08317.
 - [39] Zhou Haoyi, Zhang Shanghang, Peng Jieqi et al. *Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting*. — 2021. — 2012.07436.
 - [40] FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting / Tian Zhou, Ziqing Ma, Qingsong Wen et al. // Proc. 39th International Conference on Machine Learning (ICML 2022). — 2022.
 - [41] Markowitz Harry. *Portfolio Selection // The Journal of Finance*. — 1952. — Vol. 7, no. 1. — P. 77–91.
 - [42] Sharpe William F. *Capital asset prices: A theory of market equilibrium under conditions of risk // The Journal of Finance*. — 1964. — Vol. 19, no. 3. — P. 425–442.
 - [43] Oladejo Nathaniel, Abolarinwa Abimbola, Salawu Sulyman. *Linear Programming and Its Application Techniques in Optimizing Portfolio Selection of a Firm // Journal of Applied Mathematics*. — 2020. — 12. — Vol. 2020. — P. 1–7.
 - [44] Farid Fajri, Rosadi Dedi. *Portfolio optimization based on self-organizing maps clustering and genetics algorithm // International Journal of Advances in Intelligent Informatics*. — 2022. — 03. — Vol. 8. — P. 33.
 - [45] Lang Jonas, Zielinski Sebastian, Feld Sebastian. *Strategic Portfolio Optimization Using Simulated, Digital, and Quantum Annealing // Applied Sciences*. — 2022. — 12. — Vol. 12. — P. 12288.
 - [46] Antonov Anton. *Portfolio Optimization using Machine Learning*. — 2018. — 11.
 - [47] Boyd Stephen, Busseti Enzo, Diamond Steven et al. *Multi-Period Trading via Convex Optimization*. — 2017. — 1705.00109.

-
- [48] Sousa Lobo Miguel, Fazel Maryam, Boyd Stephen. Portfolio optimization with linear and fixed transaction // Annals OR. — 2007. — 03. — Vol. 152. — P. 341–365.
 - [49] Balch Tucker Hybinette, Mahfouz Mahmoud, Lockhart Joshua et al. How to Evaluate Trading Strategies: Single Agent Market Replay or Multiple Agent Interactive Simulation? — 2019. — 1906.12010.
 - [50] Clusterization of Indices and Assets in the Stock Market / Leszek Chmielewski, Maciej Janowicz, Luiza Ochnio, Arkadiusz Orłowski. — Vol. 9375. — 2015. — 10.
 - [51] Working Memory Connections for LSTM / Federico Landi, Lorenzo Baraldi, Marcella Cornia, Rita Cucchiara // Neural Networks. — 2021. — . — Vol. 144. — P. 334–341. — URL: <http://dx.doi.org/10.1016/j.neunet.2021.08.030>.
 - [52] Evolution of worldwide stock markets, correlation structure, and correlation-based graphs / Dong-Ming Song, Michele Tumminello, Wei-Xing Zhou, Rosario N. Mantegna // Physical Review E. — 2011. — . — Vol. 84, no. 2. — URL: <http://dx.doi.org/10.1103/PhysRevE.84.026108>.
 - [53] A survey on long short-term memory networks for time series prediction / Benjamin Lindemann, Timo Müller, Hannes Vietz et al. // Procedia CIRP. — 2021. — Vol. 99. — P. 650–655. — 14th CIRP Conference on Intelligent Computation in Manufacturing Engineering, 15-17 July 2020. URL: <https://www.sciencedirect.com/science/article/pii/S2212827121003796>.
 - [54] Arratia Argimiro, Sepúlveda Eduardo. Convolutional Neural Networks, Image Recognition and Financial Time Series Forecasting. — 2020. — 01. — P. 60–69. — ISBN: 978-3-030-37719-9.
 - [55] Eckmann Jean-Pierre, Kamphorst S O, Ruelle David. Recurrence Plots of Dynamical Systems // Europhysics Letters. — 1987. — Vol. 5, no. 9. — P. 973–977.
 - [56] Garza Azul, Mergenthaler-Canseco Max. TimeGPT-1. — 2023. — 2310.03589.

Приложение А

Перечень технических индикаторов с их описанием

Field	Type	Description	Example values
pr open	double	цена открытия	113.76
pr high	double	максимальная цена за период	113.96
pr low	double	минимальная цена за период	113.06
pr close	double	последняя цена за период	113.72
pr std	double	стандартное отклонение цены	0.0005
vol	int32	объем в лотах	1514
val	double	объем в рублях	1718129
trades	int32	количество сделок	87
pr vwap	double	взвешенная средняя цена	113.48
pr change	double	изменение цены за период, %	-0.0352
trades b	int32	кол-во сделок на покупку	45
trades s	int32	кол-во сделок на продажу	42
val b	double	объем покупок в рублях	922434
val s	double	объем продаж в рублях	795695
vol b	int64	объем покупок в лотах	812
vol s	int64	объем продаж в лотах	702
disb	double	соотношение объема покупок и продаж	0.07
pr vwap b	double	средневзвешенная цена покупки	113.6
pr vwap s	double	средневзвешенная цена продажи	113.35

Таблица A.1: TradeStats

Field	Type	Description	Example values
put orders b	int32	кол-во поставленных заявок (покупка)	1042
put orders s	int32	кол-во поставленных заявок (продажа)	1085
put val b	double	объем заявок поставленных в стакан (покупка, руб)	63241202
put val s	double	объем заявок поставленных в стакан (продажа, руб)	63241202
put vol b	int32	объем заявок поставленных в стакан (покупка)	56056
put vol s	int32	объем заявок поставленных в стакан (продажа)	53390
put vwap b	double	средневзвешенная цена заявок (покупка)	112.82
put vwap s	double	средневзвешенная цена заявок (продажа)	112.57
put vol	int32	объем заявок, поставленных в стакан	109446
put val	double	объем заявок, поставленных в стакан (руб)	123341703
put orders	int32	кол-во поставленных заявок в стакан	2127
cancel orders b	int32	кол-во снятых заявок (покупка)	983
cancel orders s	int32	кол-во снятых заявок (продажа)	1012
cancel val b	double	объем снятых заявок (покупка, руб)	61001435
cancel val s	double	объем снятых заявок (продажа, руб)	61001435
cancel vol b	int32	объем снятых заявок (покупка, лоты)	53936
cancel vol s	int64	объем снятых заявок (продажа, лоты)	51538
cancel vwap b	double	средневзвешенная цена отмененных заявок (покупка)	113.1
cancel vwap s	double	средневзвешенная цена отмененных заявок (продажа)	113.85
cancel vol	int64	объем снятых заявок	105474
cancel val	double	объем снятых заявок (руб)	119675609
cancel orders	int64	кол-во снятых заявок	1995

Таблица A.2: OrderStats

Field	Type	Description	Example values
spread bbo	double	спред между лучшей ценой покупки и продажи	13.8
spread lv10	double	спред между 10ым уровнем цен покупки и продажи	71.5
spread 1mio	double	спред на 1 млн руб	62.6
levels b	int32	кол-во уровней цен в стакане (покупка)	263
levels s	int32	кол-во уровней цен в стакане (продажа)	290
vol b	int64	совокупный объем заявок в стакане на всех уровнях (покупка)	6413
vol s	int64	совокупный объем заявок в стакане на всех уровнях (продажа)	24111
val b	int64	совокупный объем заявок в стакане на всех уровнях (покупка), руб	6917245
val s	int64	совокупный объем заявок в стакане на всех уровнях (продажа), руб	30842818
imbalance vol bbo	double	дисбаланс объема на лучших ценах	-0.04
imbalance val bbo	double	дисбаланс объема (руб) по лучшим ценам	-0.04
imbalance vol	double	дисбаланс объема на всем стакане (все уровни)	-0.04
imbalance val	double	дисбаланс объема (руб) на всем стакане (все уровни)	-0.63
vwap b	double	средневзвешенная цена покупки в стакане	107.85
vwap s	double	средневзвешенная цена продажи в стакане	127.92
vwap b 1mio	double	цена покупки актива на 1 млн руб	113.19
vwap s 1mio	double	цена продажи актива на 1 млн руб	113.9

Таблица А.3: OBStats