



МИНОБРНАУКИ РОССИИ
федеральное государственное бюджетное образовательное
учреждение высшего образования
«Национальный исследовательский университет «МЭИ»

Институт
Кафедра

ИВТИ
ПМИИ

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
(БАКАЛАВРСКАЯ РАБОТА)

Направление 01.03.02 Прикладная математика и информатика
(код и наименование)

Образовательная программа Математическое и программное обеспечение
вычислительных машин и компьютерных сетей

Форма обучения очная
(очная/очно-заочная/заочная)

Тема: Применение нейронных сетей для прогнозирования финансовых
временных рядов

Студент А-136-20 Имамкулов Х.М.
группа подпись фамилия и инициалы

Руководитель ВКР К.Т.Н. доцент Кожевников А.В.
уч. степень должность подпись фамилия и инициалы

Консультант
уч. степень должность подпись фамилия и инициалы

Внешний консультант
уч. степень должность подпись фамилия и инициалы

организация

«Работа допущена к защите»

Заведующий кафедрой К.Т.Н. доцент Варшавский П.Р.
уч. степень звание подпись фамилия и инициалы

Дата

Москва, 2024



МИНОБРНАУКИ РОССИИ
федеральное государственное бюджетное образовательное
учреждение высшего образования
«Национальный исследовательский университет «МЭИ»

Институт
Кафедра

ИВТИ
ПМИИ

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
(БАКАЛАВРСКУЮ РАБОТУ)

Направление 01.03.02 Прикладная математика и информатика
(код и наименование)

Образовательная программа Математическое и программное обеспечение
вычислительных машин и компьютерных сетей

Форма обучения очная
(очная/очно-заочная/заочная)

Тема: Применение нейронных сетей для прогнозирования финансовых
временных рядов

Студент А-136-20 Имамкулов Х.М.
группа подпись фамилия и инициалы

Руководитель ВКР К.Т.Н. доцент Кожевников А.В.
уч. степень должность подпись фамилия и инициалы

Консультант
уч. степень должность подпись фамилия и инициалы

Внешний консультант
уч. степень должность подпись фамилия и инициалы

Заведующий кафедрой организация
К.Т.Н. доцент Варшавский П.Р.
уч. степень звание подпись фамилия и инициалы

Место выполнения работы ФГБОУ ВО «НИУ «МЭИ»

СОДЕРЖАНИЕ РАЗДЕЛОВ ЗАДАНИЯ И ИСХОДНЫЕ ДАННЫЕ

1. Исследование специфики анализа временных рядов с помощью методов машинного обучения.
2. Выбор и обучение моделей машинного обучения для анализа временных рядов цен акций.
3. Разработка и реализация системы для прогнозирования финансовых временных рядов.
4. Анализ полученных результатов

ПЕРЕЧЕНЬ ГРАФИЧЕСКОГО МАТЕРИАЛА

Количество листов

Количество слайдов в презентации

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

- 1.Текст: электронный // Studfile.net. — URL: <https://studfile.net/preview/3320299/page:14/>
- 2.Текст:электронный//https://www.galaktika-dmk.com/upload/iblock/001/7ilgv6zyfufx33uxwoemzn19m1jvz3z2/45582-_978_5_93700_212_9_.pdf
- 3.Текст:электронный//Электронный архив УрФУ. — URL: <https://elar.urfu.ru/handle/10995/52487>
4. Текст: электронный // NeuralProphet. — URL: <https://neuralprophet.com/>
5. Текст: электронный // Loginom. — URL: <https://loginom.ru/blog/arima-example>

Аннотация

Выпускная квалификационная работа выполнена Имамкулов Хайдаром Махмадзакировичем, студентом группы А-136-20 ФГБОУ ВО «Национальный исследовательский университет «Московский энергетический институт» в 2024 году.

Тема работы: «Применение нейронных сетей для прогнозирования финансовых временных рядов».

Расчетно-пояснительная записка занимает 72 страницы (54 без приложений), содержит 21 иллюстрацию, 2 таблицы и 4 приложения. Список использованных источников включает в себя 13 наименований.

Работа посвящена исследованию и выбору наилучшего метода для прогнозирования цен акций на бирже, а также разработке приложения, предсказывающего будущие значения акций на заданный период.

Оглавление

Аннотация	4
ВВЕДЕНИЕ	7
Глава 1. Нейронные сети для прогнозирования	8
1.1 Формализация задачи прогнозирования	8
1.2 Нейронные сети	9
1.2.1 Характеристики нейронных сетей	10
1.2.2 Структура нейронной сети	10
1.2.3 Область применения нейронных сетей	12
1.2.4 Пример использования	13
1.3 Сравнительный анализ методов машинного обучения для прогнозирования финансовых временных рядов	13
1.4 Оценка качества моделей	20
1.5 Вывод по первой главе	21
Глава 2. Применение нейронных сетей для анализа финансовых временных рядов	22
2.1 Задача прогнозирования	22
2.2 Прогнозирование с помощью сведения к задаче регрессии	23
2.3 Отбор признаков	24
2.3.1 Сезонность	25
2.3.2 Счетчики	26
2.4 Построение прогноза	27
2.5 Кросс-валидация для временных рядов	30
2.6 Neural Prophet.	34
2.7 Вывод по второй главе	36
Глава 3. Обучение и выбор наилучшей модели.	37
3.1 Подготовка данных	37
3.2. Преобразование данных	39
3.3 Инициализация и обучение модели	40
3.4 Выбор наилучшей модели	45
3.5 Вывод по третьей главе	46
Глава 4. Программная реализация системы прогноза значение акций на бирже	47
4.1. Описание программной части	47
4.2. Тестирование разработанной программы	48
4.3 Вывод по четвертой главе	51
ЗАКЛЮЧЕНИЕ	53

Список литературы.....	54
Приложение 1	56
Приложение 2	63
Приложение 3	70
Приложение 4	72

ВВЕДЕНИЕ

Целью настоящей работы является разработка и реализация, а также анализ эффективности применения нейронных сетей для прогнозирования финансовых временных рядов. Финансовые временные ряды играют ключевую роль в современной экономике и финансовых рынках. Анализ и прогнозирование этих рядов позволяют участникам рынка принимать обоснованные решения, минимизировать риски и увеличивать доходность инвестиций. В последние десятилетия прогнозирование финансовых временных рядов стало одной из самых актуальных задач в финансовой аналитике

Первая глава работы посвящена нейронным сетям – инструменту машинного обучения, широко используемому в анализе данных и прогнозировании. Нейронные сети являются одним из наиболее эффективных инструментов предсказательной аналитики и интеллектуального анализа данных, позволяя решать задачи классификации, регрессии и прогнозирования временных рядов. Существуют различные архитектуры нейронных сетей, каждая архитектура имеет свои особенности и применяется в зависимости от специфики задачи. Во второй главе рассматриваются основные архитектуры нейронных сетей, используемые для прогнозирования временных рядов, а также методы предварительной обработки данных и настройки гиперпараметров для улучшения точности моделей. Третья глава посвящена обучению и выбору наилучшей модели, данный этап включает в себя обработку исходных данных и преобразованию их в временной ряд, а также настройки параметров выбранной модели и проверки точности прогнозов. В четвертой главе рассмотрена работа с реализованным приложением, включающим в себя наилучшую из моделей, вводимый набор данных и количество дней, на которых будет строиться прогноз.

Глава 1. Нейронные сети для прогнозирования

В главе описывается метод нейронных сетей, указывается их структура, приведен сравнительный анализ с другими методами машинного обучения.

1.1 Формализация задачи прогнозирования

Задача прогнозирования состоит в формировании модели, способной предсказать будущие значения состояния системы или объекта на основе исторических данных [1].

В самом общем виде задача прогнозирования может быть сформулирована в терминах «условия – прогноз». Условия включают множество исторических состояний системы и множество параметров модели. Иными словами, задание условий представляется формальным описанием средств и результатов. Прогноз же определяет ожидаемое состояние системы в будущем. Многообразие различных практических задач прогнозирования обуславливает многообразие детализаций этой общей постановки.

В данной работе рассматривается задача прогнозирования цен акций, для которой необходимо по известным параметрам временного ряда определить будущую цену акций. При прогнозировании каждая единица наблюдения соответствует определенному временному интервалу, на основе которого делается прогноз на следующий интервал (рис.1).

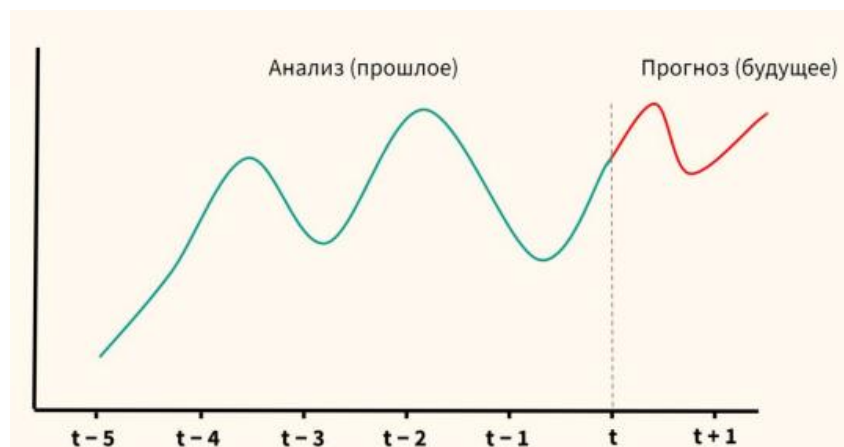


Рис.1 Исторические данные и их прогноз

1.2 Нейронные сети

Нейронная сеть – метод представления сложных нелинейных зависимостей в иерархической структуре, состоящей из входного слоя, одного или нескольких скрытых слоев и выходного слоя. В узлах скрытых слоев хранятся веса и функции активации, которые используются для вычисления выходных значений. Целью нейронных сетей является создание модели, которая предсказывает значение целевой переменной на основе нескольких входных переменных. В самом простом случае множество входных значений проходит через несколько слоев, чтобы получить прогноз на выходе [2].

Каждый слой сети состоит из нейронов, которые выполняют вычисления и передают результаты на следующий слой. Обучение сети осуществляется за счет настройки весов связей между нейронами на основе исторических данных, которые позволяют модели выявлять зависимости и делать точные прогнозы.

Нейронная сеть в качестве входных данных принимает множества анализируемых атрибутов, и возвращает “прогноз”, который представляет

предсказанный выходной набор атрибутов, соответствующий входным данным. Входные же атрибуты могут быть как дискретными, так и непрерывными.

1.2.1 Характеристики нейронных сетей

В данном разделе описаны основные характеристики нейронных сетей.

Ширина нейронной сети – наибольшее количество нейронов на одном слое нейронной сети.

Глубина нейронной сети – количество слоев нейронов или максимальное количество последовательно соединенных слоев.

Количество нейронов – общее количество нейронов в нейронной сети.

От данных характеристик зависит эффективность нейронной сети для прогнозирования финансовых временных рядов, включая скорость обучения и точность прогнозирования. Чем больше нейронов и глубина сети, тем больше способность модели выявлять сложные зависимости во временных рядах.

1.2.2 Структура нейронной сети

Существует множество различных архитектур нейронных сетей, которые эффективны для различных областей применения, однако существуют две основные категории структур нейронных сетей: ациклические и рекуррентные сети

Ациклические нейронные сети

Персептрон — это один из простейших типов нейронных сетей, который состоит из одного или нескольких слоев нейронов без циклических связей. Он хорошо справляется с задачами классификации и регрессии, но его возможности

ограничены, когда речь идет о временных рядах, так как он не учитывает зависимость данных во времени.

Сверточные нейронные сети (CNN) — эти сети изначально разрабатывались для обработки изображений и данных с пространственной структурой, но они также могут быть использованы для работы с временными рядами. В финансовом анализе сверточные сети могут применяться для выявления локальных закономерностей и особенностей в данных. CNN могут извлекать важные признаки из временных рядов, которые затем могут быть использованы в других моделях для прогнозирования[\[3\]](#).

Рекуррентные нейронные сети

Рекуррентные нейронные сети (RNN) — это класс нейронных сетей, специально предназначенный для работы с последовательными данными. В отличие от ациклических сетей, RNN имеют петли, позволяющие информации сохраняться и передаваться от одного шага последовательности к другому. Это делает их особенно подходящими для задач, где важно учитывать временные зависимости, таких как прогнозирование финансовых временных рядов.

Однако стандартные RNN имеют свои ограничения, что затрудняет обучение на длинных последовательностях данных. Для решения этих проблем были разработаны более продвинутые архитектуры: долгосрочная краткосрочная память (LSTM), управляемые рекуррентные блоки (GRU), которые рассмотрены в разделе 1.3.

1.2.3 Область применения нейронных сетей

Нейронные сети широко применяются для анализа и прогнозирования в различных областях. В медицине они используются для прогнозирования распространения заболеваний и мониторинга состояния пациентов. В энергетике они прогнозируют потребление электроэнергии и управляют выработкой возобновляемых источников энергии. В транспорте и логистике нейронные сети прогнозируют трафик и управляют запасами, что позволяет оптимизировать цепочки поставок и маршруты.

Также они используются во множестве финансовых задач для прогнозирования временных рядов и анализа рынка. Применение нейронных сетей в этом ключе включает в себя:

- Прогнозирование цен акций на основе исторических данных цен акций и других финансовых показателей. В данной работе основное внимание уделяется именно этой задаче.
- Прогнозирование валютных курсов для определения будущих изменений курсов валют и торговых операций на валютном рынке.
- Анализ временных рядов для выявления трендов, сезонности и цикличности в финансовых данных.
- Управление портфелем для принятия решений о распределении активов в портфеле и оптимизации инвестиционных стратегий.

1.2.4 Пример использования

В качестве примера можно рассмотреть задачу прогнозирования будущих цен акций компании на основе ежедневных данных о ценах за последние несколько лет, а также других факторов, таких как объем торгов, финансовые показатели компании и макроэкономические индикаторы.

Для начала собираются исторические данные о ценах акций X за последние несколько лет. Полученные данные затем проходят процесс предобработки, включающий очистку данных от пропущенных значений и обработку выбросов.

Поскольку значения цен за некоторый период времени зависимы друг от друга, для решения данной задачи имеет смысл представить исторические данные в виде временных рядов. Затем, выбрав определенный период времени (например, последние 10 дней), можно сформировать обучающие примеры, где входы представляют собой исторические данные за определенный период времени, а выходом является цена акции на следующий день.

На сформированных примерах обучается модель, которая после завершения обучения, может использоваться для прогнозирования цен акций на основе текущих данных о последних ценах акций, объемах торгов и других факторах. Полученные прогнозы могут быть использованы для поддержки инвестиционных решений, таких как покупка, продажа или удержание акций X , основанных на предполагаемых движениях цен.

1.3 Сравнительный анализ методов машинного обучения для прогнозирования финансовых временных рядов

Поскольку данная работа посвящена анализу финансовых временных рядов цен акций на бирже, в работе рассматриваются наиболее распространенные методы

машинного обучения для прогнозирования временных рядов. Каждый из этих методов имеет свои преимущества и недостатки, и выбор конкретного метода зависит от характеристик данных, структуры временного ряда и целей анализа.

1. ARIMA (авторегрессивное интегрированное скользящее среднее)

ARIMA — это модель временных рядов, которая объединяет в себе три основных компонента:

- **Авторегрессия (AR):** этот компонент модели описывает зависимость текущего значения ряда от его предыдущих значений.
- **Интегрирование (I):** этот компонент используется для стабилизации временного ряда путем вычитания значения ряда из его предыдущего значения.
- **Скользящее среднее (MA):** этот компонент описывает зависимость текущего значения ряда от шума и ошибок предыдущих периодов.

ARIMA модели подходят для прогнозирования временных рядов с явными трендами и сезонностью (рис.2)[\[4\]](#).

$$\begin{array}{ccccccc}
 (1 - \phi_1 B) & (1 - \Phi_1 B^4) & (1 - B) & (1 - B^4) y_t = & (1 + \theta_1 B) & (1 + \Theta_1 B^4) e_t. \\
 \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow \\
 \left(\begin{array}{c} \text{Non-seasonal} \\ \text{AR}(1) \end{array} \right) & & \left(\begin{array}{c} \text{Non-seasonal} \\ \text{difference} \end{array} \right) & & \left(\begin{array}{c} \text{Non-seasonal} \\ \text{MA}(1) \end{array} \right) & & \\
 & \left(\begin{array}{c} \text{Seasonal} \\ \text{AR}(1) \end{array} \right) & & \left(\begin{array}{c} \text{Seasonal} \\ \text{difference} \end{array} \right) & & \left(\begin{array}{c} \text{Seasonal} \\ \text{MA}(1) \end{array} \right)
 \end{array}$$

Рис.2.Формула описывающая модель ARIMA

Достоинства:

- Модели такого типа легко интерпретировать и объяснить.
- Может быть применена к различным типам временных рядов, включая ряды с явными трендами и сезонностью.
- Может обрабатывать ряды с большим объемом данных и демонстрировать хорошие результаты прогнозирования.

Недостатки:

- Может давать различные прогнозы в зависимости от выбранных начальных параметров модели.
- Модель требует, чтобы временной ряд был *стационарным* (это свойство временного ряда, которое означает, что его средние и стандартные отклонения не меняются со временем), что может потребовать дополнительных преобразований данных.
- ARIMA не всегда хорошо работает с внешними факторами, влияющими на временной ряд.

2. EWMA (экспоненциально взвешенное скользящее среднее)

EWMA — это метод сглаживания временных рядов, который уделяет больше внимания последним наблюдениям, чем более ранним. В этом методе каждое новое наблюдение вносит больший вклад в среднее значение, чем предыдущие наблюдения. Это позволяет модели быстрее реагировать на изменения в данных[\[5\]](#).

В общем виде метод EWMA можно описать формулой:

$$EWMA_t = a * r_t + (1 - a) * EWMA_{t-1}, \text{ где}$$

- $EWMA$ -скользящее среднее в момент времени t ,
- a - степень смешивания значений параметров между 0 и 1,
- r - значение r в момент времени t ,

Достоинства:

- $EWMA$ прост в реализации и требует минимум параметров.
- быстро реагирует на изменения в данных, что делает его хорошим выбором для анализа временных рядов с частыми колебаниями.

Недостатки:

- эффективность зависит от правильного выбора параметра сглаживания.
- может не обнаруживать долгосрочные зависимости в данных из-за его ориентации на последние наблюдения.

3. Динамическая регрессия

Динамическая регрессия — это метод, который учитывает влияние экзогенных переменных (внешних факторов) на временной ряд. В этом методе временной ряд моделируется с использованием регрессионной модели, включающей в себя как автокорреляцию, так и экзогенные переменные[\[6\]](#).

$$y_t = a + \beta_0 x_t + \beta_1 x_{t-1} + \dots + \beta_k x_{t-k} + \varepsilon_t,$$

где a свободный член, коэффициенты β называют краткосрочным мультипликатором, x_t значение исходных данных, ε_t погрешность.

Достоинства:

- динамическая регрессия позволяет учитывать влияние экзогенных переменных на временной ряд.
- модель динамической регрессии может быть адаптирована для работы с различными типами данных и структурами временных рядов.

Недостатки:

- может быть сложной для интерпретации из-за большого числа включенных переменных.
- требуется наличие достаточного количества данных для построения адекватной модели регрессии.

4. LSTM (Long Short-Term Memory)

LSTM — это тип рекуррентной нейронной сети, специально разработанный для анализа и прогнозирования временных рядов. LSTM обладает способностью запоминать долгосрочные зависимости в данных, что делает его особенно эффективным для анализа временных рядов с долгосрочными зависимостями или нестационарными данными(рис.3)[\[7\]](#).

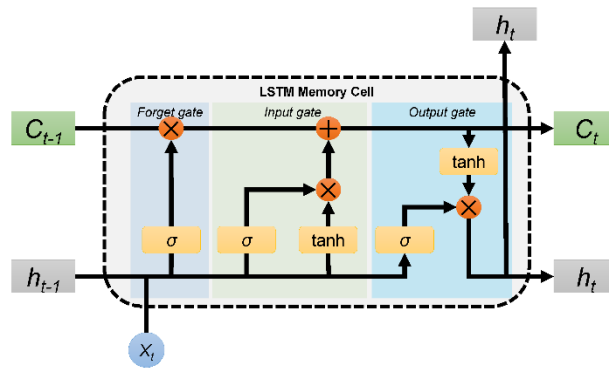


Рис.3.Схема ячейки LSTM

Достоинства:

- способен обрабатывать долгосрочные зависимости в данных, делая его эффективным для анализа временных рядов с долгосрочной динамикой.
- LSTM может автоматически выделять и использовать значимые признаки из данных.

Недостатки:

- обучение модели может требовать большого объема вычислительных ресурсов и времени.
- модели сложно интерпретировать из-за их сложной структуры и большого числа внутренних параметров.

5. GRU (Gated Recurrent Unit)

Управляемые рекуррентные блоки (GRU) — это еще одна разновидность RNN, которая была предложена как более простая и вычислительно эффективная альтернатива LSTM. Объединяет входные и забывающие механизмы в единый

обновляющийся механизм, что уменьшает количество параметров и упрощает процесс обучения(рис.4)[\[7\]](#).

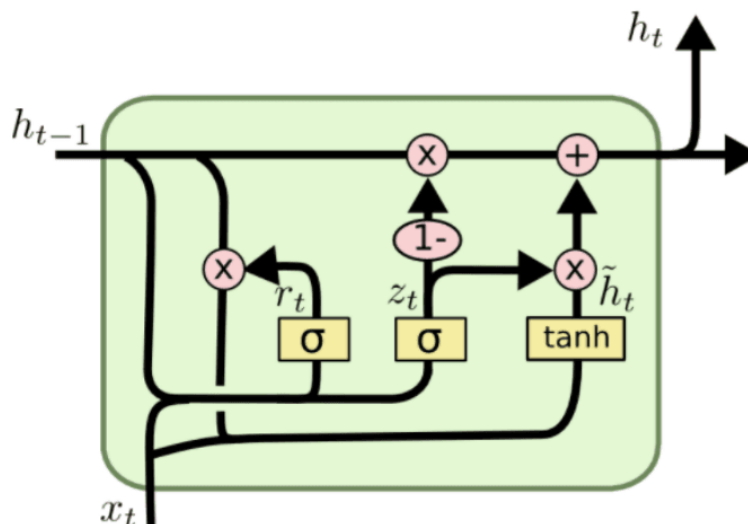


Рис.4 Схема ячейки GRU

Достоинства:

- Имеют более простую архитектуру по сравнению с LSTM, что уменьшает количество параметров, необходимых для обучения модели и снижает вычислительные затраты.
- Быстрое обучение и адаптация к новым данным за счет уменьшенного числа параметров.
- Демонстрируют высокую эффективность в задачах прогнозирования последовательных данных, включая финансовые временные ряды.

Недостатки:

- Могут быть менее гибкими и адаптируемыми в некоторых сложных задачах по сравнению с LSTM.

- В некоторых задачах может быть трудно определить, какая архитектура лучше подойдет — GRU или LSTM, что требует дополнительных экспериментов и сравнений.

1.4 Оценка качества моделей

Для оценки качества моделей прогнозирования временного ряда в основном используются следующие метрики качества регрессии[\[8\]](#).

средняя квадратичная ошибка

$$MSE = \frac{1}{T - R + 1} \sum_{t=R}^T (\hat{y}_t - y_t)^2$$

средняя абсолютная ошибка

$$MAE = \frac{1}{T - R + 1} \sum_{t=R}^T |\hat{y}_t - y_t|,$$

где T - количество наблюдений в ряду, R количество точек, y_t фактическое значение временного ряда, \hat{y}_t предсказанное значение

Эти метрики показывают абсолютное значение ошибки, на сколько рублей или на сколько единиц товара может отличаться спрогнозированный результат от реального. Также могут использоваться и относительные метрики:

средняя абсолютная ошибка в процентах

$$MAPE = \frac{100}{T - R + 1} \sum_{t=R}^T \left| \frac{\hat{y}_t - y_t}{y_t} \right|$$

взвешенная средняя ошибка в процентах.

$$WAPE = 100 * \frac{\sum_{t=R}^T |\hat{y}_t - y_t|}{\sum_{t=R}^T |y_t|}$$

Эти метрики достаточно популярны поскольку, позволяют оценить качество в относительных величинах без зависимости от шкалы измерений.

1.5 Вывод по первой главе

В данной главе приведена формализованная постановка задачи прогнозирования, указаны основные архитектуры и области применения нейронных сетей, проведен сравнительный анализ методов машинного обучения для прогнозирования финансовых временных рядов. Каждый метод имеет свои преимущества и недостатки, и выбор конкретного метода зависит от характеристик данных и целей анализа.

Глава 2. Применение нейронных сетей для анализа финансовых временных рядов

В данной главе рассматривается применение нейронных сетей для анализа финансовых временных рядов. Обсуждаются обобщенные методы построения нейронных сетей, а также специфические алгоритмы и техники, применяемые в контексте анализа финансовых данных, описывается задача прогнозирования финансовых временных рядов.

2.1 Задача прогнозирования

Предполагается, что имеются данные о значении временного ряда в различные моменты времени, тогда задача состоит в том, чтобы построить модель, которая сможет предсказывать будущие значения этого ряда(рис.5). Если имеется несколько признаков, нет необходимости прогнозировать каждый из них, обычно выделяются целевые признаки, для которых строится прогноз, а остальные признаки могут быть использованы в качестве опорных данных для прогнозирования.

Прогнозирование временных рядов

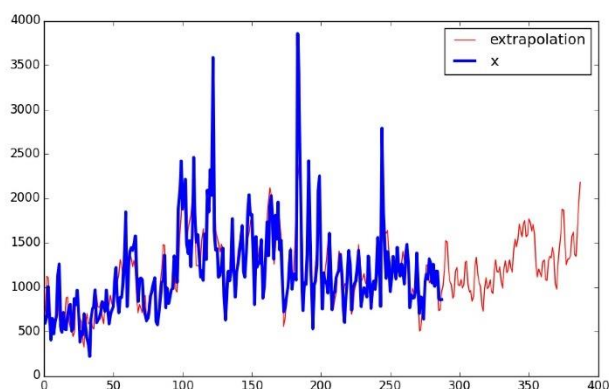


Рис.5.Исходные значения и прогнозируемые

Для этого необходимо построить некоторую функцию прогнозирования, которая на вход получает предыдущие значения ряда и параметр h , определяющий количество шагов вперед для прогноза. Таким образом, прогноз для момента времени $t+h$ строится на основе данных до момента времени t .

Чтобы оценить качество полученного прогноза, строятся предсказательные интервалы, то есть диапазоны значений, в которых с некоторой вероятностью будет лежать будущее значение ряда.

Также стоит отметить, что для повышения качества прогнозирования, рекомендуется адаптировать прогноз с течением времени, поскольку новые полученные данные могут отличаться от спрогнозированных, в таком случае следует использовать их для корректировки прогноза в качестве обновленных входных данных

2.2 Прогнозирование с помощью сведения к задаче регрессии



Рис.6. Пример временного ряда

Пусть имеются данные собранные за четыре года с 2013 по 2016 включительно(рис.6), задача прогнозирования состоит в том, чтобы на основе этих исторических данных предсказать будущие значения ряда на 2017.

Основная идея заключается в том, чтобы подать известные значения ряда в заранее сформированную (обученную на данных) функцию регрессии и получить прогноз. Модель будет иметь следующий вид:

$$y_t = f(y_{t-1}, y_{t-2}, y_{t-3}, \dots, y_{t-p})$$

где y_t предсказанное значение ряда в момент времени t , p обозначает количество предыдущих значений, а f – сформированная функция. Эту функцию можно построить, используя методы машинного обучения, такие как линейная регрессия, решающие деревья, градиентный бустинг или нейронные сети (включая сверточные и рекуррентные).

Для построения наилучшего прогноза, необходимо грамотно выбирать наиболее значимые признаки из всех имеющихся, так как они имеют весовое значение в рамках построения временного ряда.

2.3 Отбор признаков

При выборе анализируемых признаков для модели полезно начинать с визуализации временного момента t и определения доступных данных на этот момент времени. Модель может включать любые признаки, которые можно получить к моменту t , однако стоит учитывать, что часть данных может быть недоступна из-за задержек обновления, например, если данные обновляются раз в сутки

В то же время, использовать все предыдущие значения ряда в качестве признаков не всегда целесообразно: например, если история хранится годами, а прогнозировать необходимо на час вперед. Также стоит агрегировать собранные данные, например, в случаях, когда данные поступают ежесекундно, а точность требуется в минутах или часах, иначе придется учитывать неимоверно большое количество признаков.

Дополнительно к этому, стоит учитывать горизонт времени для прогноза. Например, при прогнозировании продаж для планирования поставок следует учитывать время, требуемое на сбор товаров, доставку и расстановку на полках магазина. Этот процесс может занимать от нескольких часов до нескольких дней, поэтому модель должна учитывать прогноз спроса на товар к моменту его появления на полках магазина(рис.7).



Рис.7 Отбор признаков

2.3.1 Сезонность

Сезонность занимает важную роль в анализе временных рядов. Если в ряде наблюдается сезонность, то полезно включить в модель сезонные признаки. Это можно сделать, например, следующими способами:

Значение переменной сутки/недели/месяца/года назад: Включение в модель значений переменной в тот же период времени в прошлом году или в предыдущие годы может помочь учесть сезонные колебания. Например, для прогнозирования роста или падения в определенный день можно использовать значение цены в этот же день год назад.

Сезонность, полученная методами декомпозиции ряда: существуют различные методы декомпозиции временных рядов, такие как аддитивная и мультипликативная декомпозиция, которые позволяют выделить сезонные компоненты ряда. Эти компоненты могут быть включены в модель как дополнительные признаки.

Примеры использования сезонных признаков:

1. Значение цены акций год назад: Этот признак позволяет учесть сезонные изменения цены акций в тот же период времени предыдущего года.
2. Среднее значение цены акций 23 ноября за 5 последних лет: Этот признак учитывает сезонность в цене акций в конкретный день года за последние пять лет.

2.3.2 Счетчики

В контексте прогнозирования финансовых временных рядов использование счетчиков может значительно улучшить предсказательную модель на основе нейронных сетей. Вместо того чтобы просто включать в модель исторические значения финансовых показателей, можно группировать данные по различным категориям и использовать их в качестве дополнительных факторов.

Например, можно сгруппировать данные по торговым дням с высокой и низкой волатильностью и использовать средние значения финансовых показателей для каждой группы как признаки. Также полезно рассмотреть влияние таких факторов, как объем торгов, рыночная капитализация или дивидендные выплаты, на изменение цен ценных бумаг.

Допустим, необходимо построить прогноз изменения цены акций в апреле. В этом случае стоит рассмотреть среднюю цену акций в дни с высоким и низким объемом торгов в апреле за последние несколько лет. Также можно включить в модель другие важные факторы, такие как изменения процентных ставок, экономические показатели или значимые новостные события, которые могут влиять на рынок.

Использование счетчиков таким образом помогает учесть различные аспекты финансовых временных рядов и значительно повысить точность прогнозирования с помощью нейронных сетей.

На основе вышесказанного, можно подвести итог, что при прогнозировании используются только данные из прошлого и необходимо учитывать возможные задержки в поступлении данных. Важно иметь в виду, что большое количество признаков может повысить вычислительную нагрузку модели.

2.4 Построение прогноза

После выбора признаков, необходимо выбрать, каким способом будет осуществляться прогнозирование, например, если требуется построить прогноз на несколько шагов вперед.

- **Рекурсивная стратегия**

Для каждого момента времени $t_0 \leq t \leq T$ создается объект обучающей выборки:

- *Признаковое описание* – история ряда до момента времени $t-1$.
- *Целевая метка* – значение y_t .

По этим данным обучается некоторая регрессионная модель, способная строить прогнозы на один шаг вперед. При построении прогноза на несколько шагов вперед необходимо сначала построить прогноз на один шаг, затем – на второй шаг, используя полученный прогноз на первый шаг в качестве признаков, и далее.

Иначе говоря, если для прогнозирования y_t признаковое описание имеет вид $(y_{t-1}, y_{t-2}, y_{t-3}, \dots, y_{t-p})$, то для построения прогноза y_{t+1} рассматривается признаковое описание $(y_t, y_{t-1}, y_{t-2}, y_{t-3}, \dots, y_{t-p+1})$.

На рисунке 8 изображен пример использования уже спрогнозированного значения M и $M+1$ для прогнозирования следующего ($M+1$ и $M+2$ соответственно).

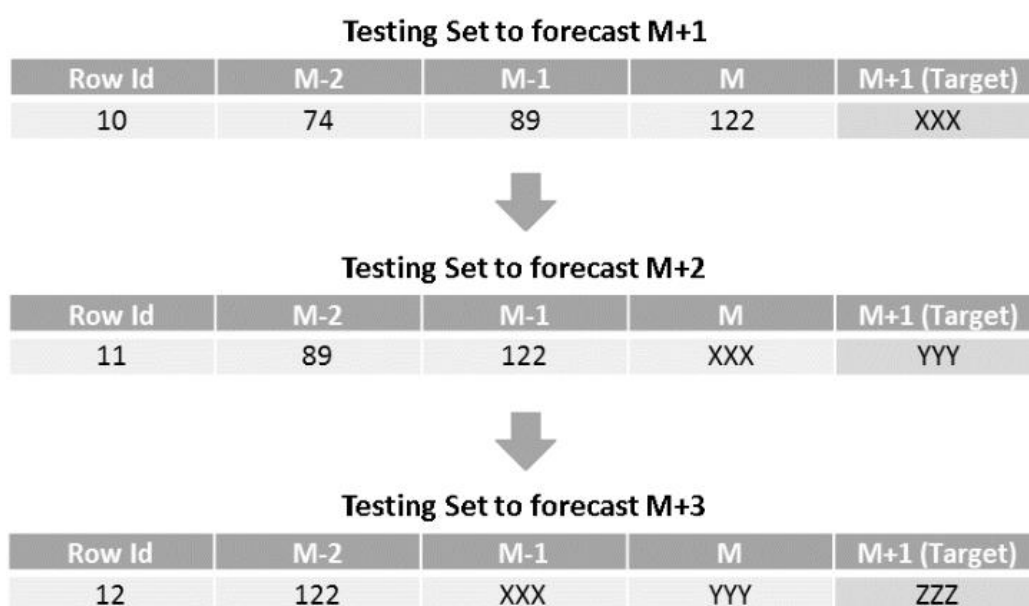


Рис.8. Пример рекурсивной стратегии

- **Прямая стратегия**

В прямой стратегии предполагается, что построением каждого прогноза в рамках горизонта прогнозирования должна заниматься своя модель. Тем самым создается N моделей прогнозирования для каждого момента времени $t_0 \leq t \leq t_0 + N - 1$.

- *Признаковое описание* – история ряда до момента времени $t_0 - 1$, причем признаки одни и те же для каждой модели.

- *Целевая метка* – значение y_t .

- **Гибридная стратегия**

Гибридная стратегия объединяет в себе преимущества рекурсивной и прямой стратегий. Как и в прямой стратегии, создается N моделей прогнозирования, но при этом каждая следующая модель использует прогнозы предыдущей подобно тому, как это делает рекурсивная стратегия.

Другими словами, необходимо построить:

- модель для прогноза на 1 шаг вперед;
- модель для прогноза на 2 шага вперед, используя прогноз уже обученной модели на 1 шаг вперед в качестве признака;
- модель для прогноза на 3 шага вперед, используя прогноз уже обученных моделей на 1 и 2 шага вперед в качестве признаков;
- и так далее обучается N моделей.

2.5 Кросс-валидация для временных рядов.

Для оценки качества работы модели машинного обучения используется кросс-валидация — это метод оценки, используемый для проверки способности модели обобщать данные на новых, ранее не встречавшихся случаях. Данный метод имеет высокую эффективность за счет постепенного, изолированного обучения моделей. Однако данный метод не совсем подходит при работе с данными в виде временных рядов, так как метод нарушает временную последовательность. Для таких случаев существует два способа построить кросс-валидацию на временных рядах[\[9\]](#).

Схема 1.

Метод кросс-валидации по первой схеме представляет собой процесс, при котором модель сначала обучается на первых t значениях временного ряда $y_1 \dots y_t$, чтобы прогнозировать следующие Δt значений $y_{t+1} \dots y_{t+\Delta t}$. Затем модель переобучается уже на основе данных $y_1 \dots y_{t+\Delta t}$ для прогноза следующего интервала $y_{t+1+\Delta t} \dots y_{t+2\Delta t}$, и так далее, пока не будет достигнут конечный интервал прогноза $y_{t+(k-1)\Delta t+1} \dots y_{t+k\Delta t}$ (рис.9). На каждом этапе обучения и прогнозирования вычисляются ошибки, которые затем усредняются для оценки общей точности модели. Этот подход позволяет адаптировать модель к изменяющимся условиям временного ряда, улучшая тем самым точность долгосрочных прогнозов.

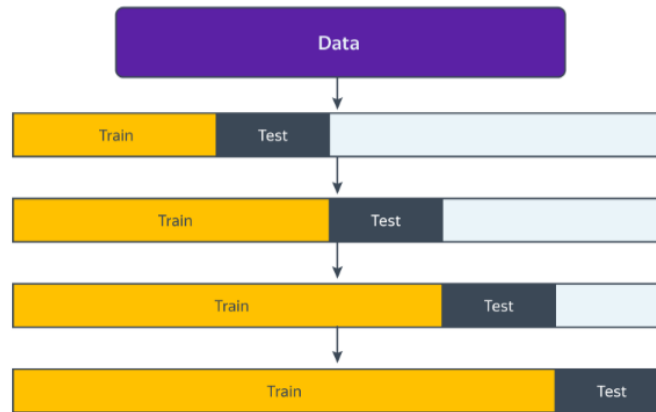


Рис.9. Пример кросс-валидации по схеме 1

Схема 2.

Процесс включает начальное обучение модели на основе первых t значений временного ряда $y_1 \dots y_t$, после чего производится прогноз на следующие Δt значений $y_{t+1} \dots y_{t+\Delta t}$, затем модель переобучается на основе последних Δt значений $y_{1+\Delta t} \dots y_{t+\Delta t}$ для прогнозирования следующего интервала $y_{t+1+\Delta t} \dots y_{t+2\Delta t}$. Этот цикл обучения и прогнозирования повторяется, каждый раз смещая диапазон обучающих данных и интервал прогноза, вплоть до достижения конечного интервала $y_{t+1+(k-1)\Delta t} \dots y_{t+k\Delta t}$ (рис.10). На каждом этапе вычисляются ошибки прогноза, которые затем усредняются для оценки общей точности модели.

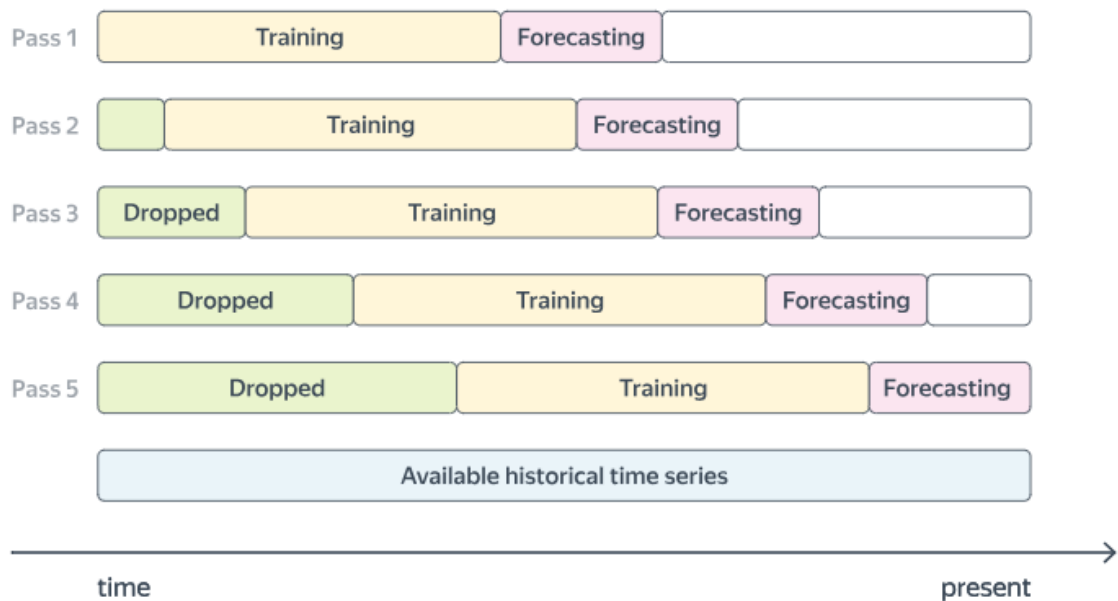


Рис.10. Пример кросс-валидации по схеме 2

Эти две стратегии отличаются лишь размером обучающего набора данных. В первом случае он постоянно увеличивается, а во втором остается неизменным, при этом сам набор данных сдвигается. Выбор конкретной стратегии зависит от поставленной задачи. Например, если имеется большое количество данных и планируется постоянное обновление модели с периодическим дообучением, то обычно при каждом дообучении размер обучающего набора фиксируют. В этом случае предпочтительно использовать вторую стратегию, чтобы оценить качество модели, обученной на фиксированном объеме данных. Если же данных недостаточно, то для обучения лучше использовать все доступные данные. В таком случае имеет смысл применить первую стратегию.

Учитывая все вышеперечисленное, стоит выделить преимущества и недостатки использования нейронных сетей в области решения задачи прогнозирования временных рядов.

Преимущества:

- Возможность использования экзогенных факторов или признаков для улучшения прогноза.
- Нейронные сети позволяют создавать много моделей для множества временных рядов одновременно. Благодаря нескольким выходам, одна модель может прогнозировать несколько рядов, например, прогнозирование продаж различных акций.

Недостатки:

- Невозможность построения предсказательных интервалов напрямую.
- Иногда показывают неточный или неправдоподобный прогноз
- Обработка признаков может быть сложной.
- Интерпретация моделей может быть затруднительна для заказчика.

2.6 Neural Prophet.

Среди существующих реализаций моделей прогнозирования временных рядов особое место занимает библиотека Facebook Prophet. Prophet был создан для работы с временными рядами, которые могут иметь сильные сезонные компоненты, пропущенные данные и нестандартные тенденции. Общая идея модели аналогична обобщенной *аддитивной модели* и учитывает такие признаки, как тенденции, сезонность, праздничные дни[\[10\]](#).

Аддитивной моделью временного ряда называется такая модель, где уровни ряда представлены как сумма трендовой, сезонной или циклической и случайной компонент, рассмотрим Facebook Prophet.

$$y(t) = g(t) + s(t) + h(t) + e(t), \text{ где}$$

- $g(t)$ относится к тренду (изменениям за длительный период времени)
- $s(t)$ относится к сезонности (периодическим или краткосрочным изменениям)
- $h(t)$ относится к влиянию праздников на прогноз
- $e(t)$ относится к безусловным изменениям, характерным для бизнеса, человека или обстоятельств (термином ошибки).

Данная модель обладает рядом достоинств и обрела высокую популярность, однако имеет ряд ограничений в виде отсутствия обработки контекста при прогнозировании и лимитированного набора регрессионных функций. Для повышения качества прогнозирования на основе данной модели разработана

библиотека NeuralProphet, которая объединяет в себе качества модели авторегрессии (Facebook Prophet) и нейронных сетей (PyTorch). Разработка данной модели позволила повысить гибкость анализа данных за счет обработки сложных закономерностей с помощью нейронных сетей[\[11\]](#).

Преимущества:

- Прост в использовании и настройке, что делает его доступным для пользователей с разным уровнем опыта.
- Автоматически обрабатывает данные с различными трендами и сезонными изменениями, позволяя пользователю задавать годовую, недельную и дневную сезонность.
- Эффективно работает с временными рядами, в которых есть пропущенные значения.
- Prophet предлагает интуитивно понятные параметры для настройки модели, что позволяет легко адаптировать ее под специфические требования временного ряда.
- Модель Prophet предоставляет результаты в удобном для интерпретации виде, что облегчает анализ прогнозов.
- Нейросетевая реализация оптимизирована под вычисления на графическом процессоре (GPU), что повышает скорость обучения и прогнозирования.

- Позволяет строить более сложные зависимости за счет нейросетевой обработки.

2.7 Вывод по второй главе

Во второй главе рассматривается специфика анализа временных рядов, описаны проблемы отбора признаков, описаны возможные стратегии при построении регрессионных моделей для прогнозирования, приведена нейросетевая модель для анализа временных рядов NeuralProphet, являющаяся нейросетевой авторегрессионной модели Facebook Prophet. Учитывая все упомянутое выше, стоит отметить, что Facebook Prophet предоставляет возможность настраивать гиперпараметры и удобный анализ результатов прогнозирования, однако он ограничен в обработке данных и выборе регрессионных функций, ввиду этих лимитов был разработан Neural Prophet, который в свою очередь, компенсирует данные недостатки за счет встроенных нейронных сетей на базе PyTorch и функций регрессии, что делает его мощным инструментом в сфере анализа временных рядов.

Глава 3. Обучение и выбор наилучшей модели.

В данной работе для подготовки модели прогнозирования финансовых временных рядов предлагается использовать:

- язык Python – наиболее популярный язык для решения задач машинного обучения, благодаря многочисленным библиотекам для ML;
- среду JupyterNotebook – в качестве среды разработки для написания кода и отображения результатов работы;
- библиотека Pandas – для работы с табличными данными: чтение, хранение, обработка, анализ и представление;
- библиотеки Matplotlib и Plotly– для создания визуализаций результатов работы в виде графиков;
- Neural Prophet – для обучения модели прогнозирования временных рядов.

3.1 Подготовка данных

Для обучения модели в качестве набора данных взята информация с сайта Kaggle, которая хранит датированную информацию о компаниях из индекса **S&P500**.

В индексе **S&P500** представлены компании из разных секторов экономики, также пакет акций является самым надежным и имеет постоянный рост:

Technology — технологические гиганты: Apple и Microsoft.

Healthcare — компании, связанные с медициной: предоставляют медицинские услуги, занимаются исследованиями в сфере биотехнологий, производят медицинское оборудование и лекарства.

Consumer Discretionary — компании из сферы потребительских товаров вторичной необходимости.

Communication Services — провайдеры мобильной связи, беспроводных и проводных услуг, медиа- и развлекательных сервисов, а также рекламные платформы, поисковики и социальные сети.

Financials — банки, инвестиционные, страховые компании, а также организации, предоставляющие услуги бизнесу и розничным клиентам.

Industrials — компании, которые производят тяжелую, сельскохозяйственную и строительную технику, станки, оборудование.

Consumer Staples — компании, предлагающие товары первой необходимости: продукты питания, напитки, товары для дома и личной гигиены.

Utilities — компании, поставляющие электричество и предоставляющие коммунальные услуги.

Загруженные данные представляют собой таблицу, где строки то наблюдения за разные даты, а столбцы различные параметры, такие как дата и значения акций. (рис.11)

	Date	ABNB	MDLZ	CI	ANET	KLAC	PANW	PLD	ADP	MMC	...	MAS	NDSN	ENPH
0	2023-06-01	112.160004	71.910957	248.869064	166.679993	448.153168	216.789993	119.265953	207.079636	171.913635	...	48.184620	219.409195	181.470001
1	2023-06-02	118.059998	72.555717	252.444122	162.500000	453.629669	217.240005	121.483223	211.684448	173.865143	...	50.551624	224.801163	181.860001
2	2023-06-05	115.690002	72.575256	258.052216	162.630005	453.619751	226.789993	121.201202	211.410690	174.614227	...	49.981968	222.159607	182.729996
3	2023-06-06	117.300003	71.588570	259.719910	159.679993	456.541229	224.720001	122.368187	211.899536	173.924316	...	52.103428	222.862015	182.610001
4	2023-06-07	116.550003	70.523727	258.328522	155.589996	459.274536	216.250000	124.536827	211.635559	172.376862	...	53.871307	230.806564	182.089996
...
247	2024-05-24	144.470001	68.300003	332.609985	306.549988	779.059998	321.600006	104.750000	248.899994	207.889999	...	69.099998	237.940002	125.180000
248	2024-05-28	147.009995	67.570000	333.209991	307.489990	786.140015	308.010010	105.000000	243.300003	203.970001	...	67.480003	233.580002	129.380005
249	2024-05-29	146.610001	66.930000	332.920013	308.309998	772.359985	306.899994	105.000000	240.089996	201.960007	...	67.330002	230.000000	125.690002
250	2024-05-30	145.520004	67.889999	331.000000	303.660004	770.130005	293.179993	107.570000	240.910004	204.399994	...	68.379997	231.300003	130.660004
251	2024-05-31	144.929993	68.529999	344.619995	297.649994	759.530029	294.910004	110.489998	244.919998	207.580002	...	69.919998	234.720001	127.900002

Рис.11. Показания исходных данных

3.2. Преобразование данных

После загрузки данных необходимо привести их к формату, удобному для дальнейшего анализа и построения модели. Основная задача на этом этапе — обеспечить правильное форматирование временных меток и заполнение пропусков в данных.

Для корректного анализа временных рядов важно, чтобы каждая временная метка была правильно интерпретирована как дата. Используется метод `pd.to_datetime()` для преобразования столбца с датами в формат `datetime`. Для удобства построения прогноза значения акций каждой компании суммируются и соотносятся с соответствующей датой.

Создается полный диапазон дат, включающий все дни в анализируемом периоде, с помощью функции `pd.date_range()`. Это позволяет определить все возможные временные метки и выявить пропущенные даты.

Для объединения данных и заполнения пропусков создаются два `DataFrame`: один с полным диапазоном дат и другой с оригинальными данными. Затем используется метод `pd.merge()` для объединения этих `DataFrame` по столбцу дат, оставляя пропуски для дней без данных.

На этом этапе подготовленные данные объединяются в единый `DataFrame` для анализа и построения модели прогнозирования. Временной ряд должен иметь столбцы, такие как дата и значение временного ряда, чтобы корректно работать с библиотекой `Prophet`.

Для корректного отображения прогноза значения акций всех 500 компаний объединяются в единый столбец 'Акции', что упрощает прогнозирование. Данные приводятся к формату, требуемому для `Prophet`, переименовывая столбцы: 'Date'

становится 'ds' (datestamp), а 'Акции' — 'y' (value)(рис.12). Этот формат является стандартным для Prophet и необходим для дальнейшей работы.

	ds	y
0	2023-06-01	85267.065580
1	2023-06-02	86785.947122
2	2023-06-03	86785.947122
3	2023-06-04	86785.947122
4	2023-06-05	86581.068192
...
331	2024-04-27	105435.500755
332	2024-04-28	105435.500755
333	2024-04-29	105919.921941
334	2024-04-30	104364.660661
335	2024-05-01	103816.433389

Рис.12. Данные после преобразования

Праздничные дни могут существенно влиять на поведение временного ряда, особенно в финансовой сфере. Prophet позволяет учитывать такие дни при построении модели. Создается DataFrame с праздничными днями, включая выходные, которые будут учтены в модели. Инициализация модели с учетом праздничных дней в США осуществляется с помощью `holidays(country_name='US')`.

3.3 Инициализация и обучение модели

После подготовки данных и определения праздничных дней следующими шагами станут инициализация и обучение модели. В Prophet предусмотрена возможность настройки различных параметров, таких как `changepoint_prior_scale`, который отвечает за контроль чувствительности модели к изменениям в тренде.

После завершения обучения модели, осуществляется прогнозирование будущих значений временного ряда. Для этого создается DataFrame с будущими датами, на основе которых и формируются прогнозы. На рисунке 13 представлен исходный график цен акций пакета S&P500.

Показание акций



Рис.13. График акций S&P500

Метод `make_future_dataframe` генерирует DataFrame, содержащий как исторические даты, так и будущие даты, на которые необходимо сделать прогноз. В данном случае следует спрогнозировать на 30 дней вперед. Метод `predict` возвращает данные с прогнозами, включающими средние значения, а также верхние и нижние доверительные интервалы.

В данной работе, выполнены две модели, отличающиеся по методам построения прогноза:

- Логический рост

Линейная регрессия используется для прогнозирования непрерывной зависимой переменной на основе одной или нескольких независимых переменных. Модель строится на линейной комбинации независимых переменных с соответствующими коэффициентами регрессии.

- Линейный рост

Логистическая регрессия используется для прогнозирования вероятности бинарного результата зависимой переменной. Она использует логическое преобразование для оценки вероятности успеха или неудачи на основе независимых переменных.

На рисунке 14 представлено сравнение результатов предсказаний моделей роста. График линейного роста, который находится слева, основанный на тренировочной выборке, довольно точно описывает поведение тренда индекса S&P500. Обученная модель учитывает большинство «переломов» и подстраивается под них. В отличие от линейной модели, график логического варианта неточно описывает поведение акций, он не учитывает резкие перемены тренда, а лишь усредняет значение и растет.

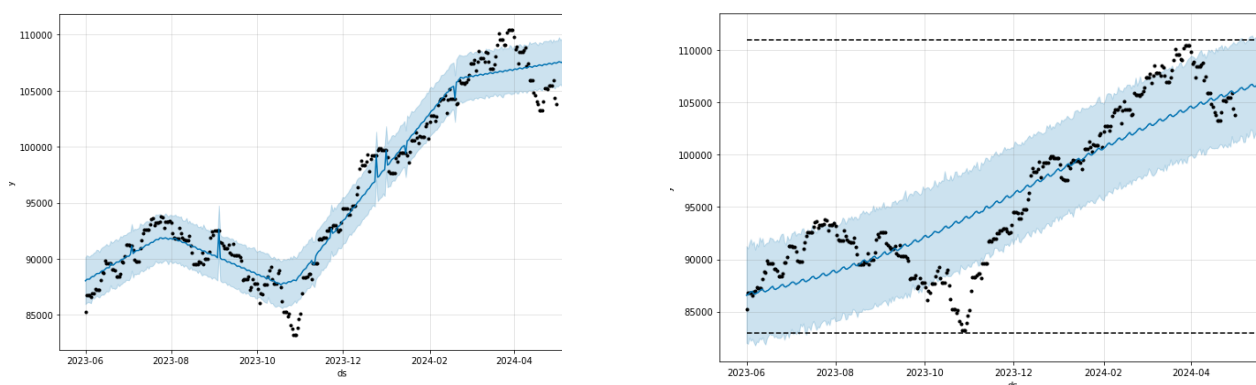


Рис.14. Графики предсказаний линейного(слева) и логического(справа) роста

Рассмотрим основные компоненты нашего временного ряда, построенного Prophet для двух типов роста (рис.15). В линейной модели (рисунок слева) хорошо описывается тренд, учитывается влияние праздничных дней, а также показания продаж акций по дням недели. Логическая же модель не учитывает тренд и выходные, что можно наблюдать по графикам компонентов.

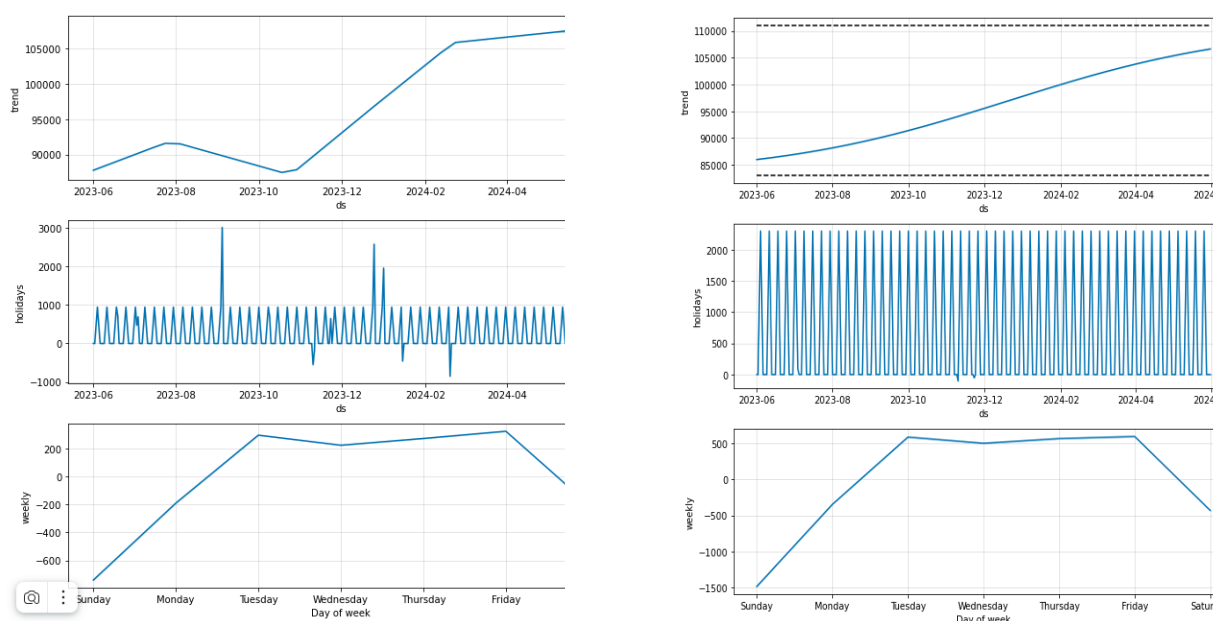


Рис.15. Графики компонентов линейного(слева) и логического роста

Для оценки качества прогнозирования по результатам работы вычислены значения средней абсолютной ошибки в процентах (MAPE) и средней абсолютной ошибки (MAE)

	MAPE	MAE
Линейный рост	0.948	1015.616
Логический рост	1.203	1300.968

На основе полученных метрик можно сделать вывод, что результаты прогноза линейной модели крайне точные, модель ошибается менее чем на 1% в процентном соотношении и на около 1016 долларов от фактического значения. Логическая модель имеет высокий уровень точности прогноза, но модель ошибается на 1.2% в процентном соотношении и на 1300 долларов от фактического значения, что уступает по показаниям этих же метрик у первого варианта.

Модель Prophet с параметром линейного роста показала отличные результаты в прогнозировании временного ряда индекса S&P 500. Высокие значения метрик точности (MAPE и MAE) свидетельствуют о том, что модель способна предсказывать значения с минимальными ошибками. Учет праздничных дней и сезонных колебаний позволяет модели учитывать важные факторы, влияющие на временной ряд, что делает её особенно эффективной для финансового прогнозирования.

После анализа стандартной модели Prophet перейдем к улучшенной Neural Prophet, данная нейронная сеть проходит тренировку на том же самом наборе данных, обучение проходит в течении 160 эпох. После обучения Neural Prophet следует можем оценить точность наших прогнозов по уже использованным метрикам MAPE, MAE.

	MAPE	MAE
Neural Prophet	0.926	993.315

Также стоит обратить внимание на график(рис.16)

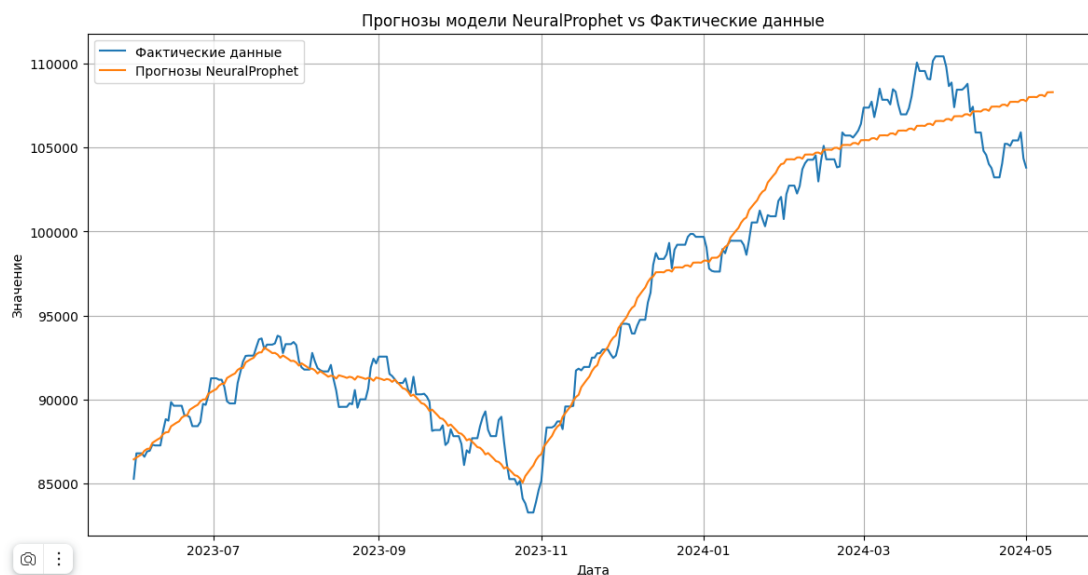


Рис.16 Прогноз Neural Prophet

Рассматриваемый график улучшенной модели Prophet, хорошо описывает тренд, в отличие от стандартной модели, выделяются области небольших «изломов», что говорит нам о чувствительности модели к резким скачкам. На прогнозируемом участке тренд имеет больший угол в отличии от линейной модели Prophet, что в свою очередь точнее прогнозирует значение акций, из-за чего метрики показали лучший результат.

3.4 Выбор наилучшей модели

Характеристики моделей:

- В первом случае (линейная модель Prophet) модель хорошо описывает тренд S&P500, учитывает праздничные дни и день недели, что способствует точным прогнозам. Ошибки модели составляют менее 1% в процентном соотношении и около 1016 долларов в абсолютном значении.
- Во втором случае (логическая модель) модель не учитывает тренд и выходные дни, что приводит к неточному описанию поведения акций. Ошибки составляют 1.203% по MAPE и 1300 долларов по MAE.

- В третьей ситуации Neural Prophet крайне точно описывает тренд индекса S&P500, за счет большей, по сравнению с линейной моделью Prophet, чувствительности. Учитывая показания метрик MAPE, MAE, которые равны 0.926 в процентном и 993.315 долларов в числовом соотношении соответственно

Оценка результатов:

- Neural Prophet демонстрирует более высокую точность с точки зрения MAPE 0.926% (против 0.948% и 1.203% в Facebook Prophet моделях) и MAE 993.315 (1015.616 долларов против 1300.968 долларов в стандартных моделях).
- Нейронная модель учитывает основные компоненты временного ряда и достаточно точно предсказывает поведение индекса, что подтверждается и метриками ошибок.

3.5 Вывод по третьей главе

В главе приводится информация о выбранном датасете S&P 500, описываются процедуры предварительной подготовки данных, получены и проанализированы результаты работы моделей Facebook Prophet и Neural Prophet.

На основе полученных результатов можно сделать вывод, что Neural Prophet является предпочтительной по сравнению с стандартной линейной и логической Prophet моделью в данном контексте. Она обеспечивает более точные прогнозы за счет учета тренда и других важных временных компонентов. Результаты ее прогнозов ближе к фактическим значениям, что делает ее более надежной для использования в анализе и прогнозировании поведения рынка.

Таким образом, для дальнейшего исследования и прогнозирования поведения акций в реализации будет использоваться Neural Prophet.

Глава 4. Программная реализация системы прогноза значение акций на бирже

4.1. Описание программной части

В рамках данной работы разработана система для прогнозирования значений акций на бирже с использованием предварительно обученной модели Prophet. Система состоит из двух основных компонентов: `main.py` и `predict.py`, которые взаимодействуют для загрузки данных, выполнения прогноза и сохранения результатов.

Файл `main.py` представляет собой скрипт, который используется для выполнения прогноза значений временных рядов с помощью предварительно обученной модели Prophet.

Для запуска системы из командной строки необходимо указать следующие аргументы:

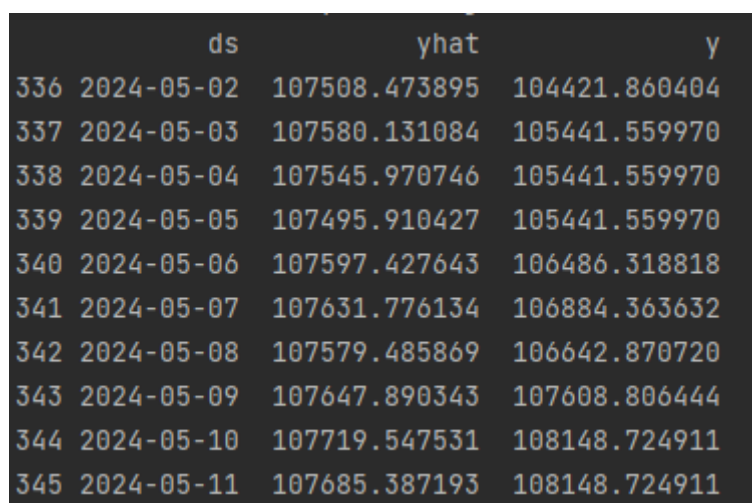
- `data_path` - путь к файлу в формате CSV с данными о временных рядах.
- `period` - количество периодов для прогноза, которые передаются в виде целого числа.

На основе полученной конфигурации система считывает данные из указанного файла, создает `DataFrame` на основе последних `period` данных. Далее загружается предварительно обученная модель `ProphetModel` из файла `final_model.json` и выполняется прогнозирование на `period` вперед на основе данных из `DataFrame`. Полученный прогноз и сохраняется в результирующий файл.

Файл `predict.py` содержит вспомогательный класс `ProphetModel`, который представляет собой обертку для работы с моделью прогнозирования временных рядов, построенной с использованием библиотеки `Prophet`.

4.2. Тестирование разработанной программы

В качестве первого примера выполним прогнозирование на 10 дней. Для начала запустим программу и установим период равный 10 дней. В результате выполнения программы ожидаем получения значения акций на период 10 дней. Программа выдает в консоль следующую информацию (рис.17).



	ds	yhat	y
336	2024-05-02	107508.473895	104421.860404
337	2024-05-03	107580.131084	105441.559970
338	2024-05-04	107545.970746	105441.559970
339	2024-05-05	107495.910427	105441.559970
340	2024-05-06	107597.427643	106486.318818
341	2024-05-07	107631.776134	106884.363632
342	2024-05-08	107579.485869	106642.870720
343	2024-05-09	107647.890343	107608.806444
344	2024-05-10	107719.547531	108148.724911
345	2024-05-11	107685.387193	108148.724911

Рис.17. Результаты прогноза на 10 дней в консоли

Рассчитаем метрику `MAPE` и `MAE` для построенного прогноза, результаты на рис.18, где «e»- `MAE`, «p» - `MAPE`

ds	e	p
2024-05-02	-3086.613491	-2.955907
2024-05-03	-2138.571114	-2.028205
2024-05-04	-2104.410776	-1.995808
2024-05-05	-2054.350457	-1.948331
2024-05-06	-1111.108825	-1.043429
2024-05-07	-747.412502	-0.699272
2024-05-08	-936.615149	-0.878273
2024-05-09	-39.083899	-0.036320
2024-05-10	429.177380	0.396840
2024-05-11	463.337718	0.428426

Рис.18. Результаты метрик

Анализируя показания программы, можно сделать вывод, что наша модель довольно точно строит прогноз,

Попробуем построить прогноз на более длительный период равный 30 дней. Запустим программу с прогнозом на 30 дней. В результате выполнения программы ожидаем получения значения акций на период 30 дней. Программа выдает в консоль следующую информацию(рис.19).

```

365 2024-05-31 108137.796874
366 2024-06-01 107632.776839
367 2024-06-02 107109.273468
368 2024-06-03 107681.650382
369 2024-06-04 108189.441923
370 2024-06-05 108137.151658
371 2024-06-06 108205.556132
372 2024-06-07 108277.213321
373 2024-06-08 107772.193286
374 2024-06-09 107248.689916
375 2024-06-10 107821.066829

```

Рис.19. Результаты прогноза на 30 дней в консоли

Прогноз уже ушел на месяц от исходных данных. Также стоит обратить внимание на все значение, которые сохранились в файле(рис.20).

,ds,yhat			
336,2024-05-02,107508.47389523123			
337,2024-05-03,107580.13108400434			
338,2024-05-04,107545.97074579127			
339,2024-05-05,107495.91042667502			
340,2024-05-06,107597.42764310424			
341,2024-05-07,107631.77613382033			
342,2024-05-08,107579.48586853285			
343,2024-05-09,107647.89034265			
344,2024-05-10,107719.54753142307			
345,2024-05-11,107685.38719321263			
346,2024-05-12,107635.32687409382			
347,2024-05-13,107736.84409052336			
348,2024-05-14,107771.19258123946			
349,2024-05-15,107718.9023159517			
350,2024-05-16,107787.30679006831			
351,2024-05-17,107858.9639788418			
352,2024-05-18,107824.80364063078			
353,2024-05-19,107774.74332151261			
354,2024-05-20,107876.2605379428			
355,2024-05-21,107910.60902865819			

Рис.20. Результаты прогноза на 30 дней в формате CSV

Далее построим график предсказанных и фактических значений для визуального сравнения(рис.21)

Прогноз цены акций

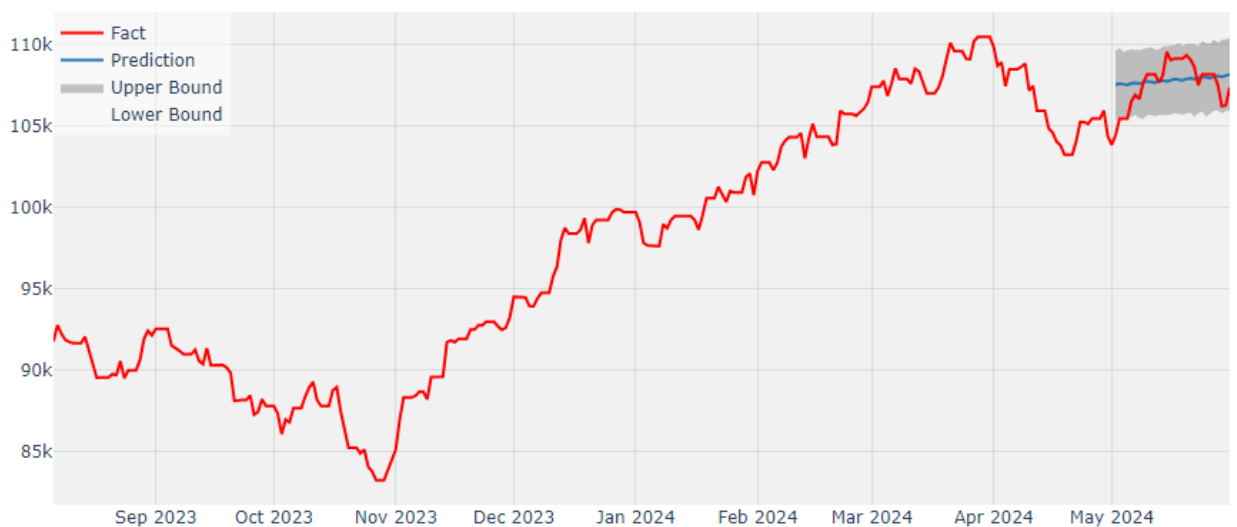


Рис.21. Фактические и прогнозируемые значения

Анализируя полученный график, можно сказать, что тренд практически не меняется, но построенный прогноз все же охватывает фактические значения.

Суммируя все вышесказанное, можно сказать, что построенная модель довольно точно предсказывает значения акций, данный тезис можно утвердить, основываясь на полученных метриках, так как их максимальное отклонение было в районе в 3%. Прогноз показывает прямой тренд, ввиду частых, но малых колебаний акций, но данный показатель построенной модели при его незначительной чувствительности, с большой точностью описывает фактические значения акций. Рассмотренная модель Neural Prophet имеет высокую точность предсказаний и соблюдает тренд, что делает из нее систему, которую можно успешно применять для решения задачи прогнозирования финансовых временных рядов цен акций.

4.3 Вывод по четвертой главе

В последней главе приводится реализация системы для прогнозирования значений акций на бирже с использованием предварительно обученной модели Prophet, указаны технические подробности реализации, рассмотрен пример работы разработанной программы. По результатам работы системы, можно сделать вывод, что обученная модель Neural Prophet достигает высокой точности в решении задачи прогнозирования финансовых временных рядов, что говорит об эффективности разработанного приложения.

ЗАКЛЮЧЕНИЕ

В результате выполнения выпускной квалификационной работы была создана программа для решения задачи прогнозирования финансовых временных рядов цен акций. В процессе работы были успешно выполнены следующие задачи:

- приведены основные определения, характеристики и архитектуры нейронных сетей, среди них выделены наиболее эффективные для решения задач прогнозирования временных рядов;
- рассмотрены различные методы по анализу временных рядов и был выбран наиболее предпочтительный из них – Neural Prophet;
- для решения указанной задачи на основе данных из индекса S&P500 были обучены три модели с использованием библиотеки Facebook и Neural Prophet линейная и логическая и нейронная сеть соответственно; на основе сравнительного анализа была выбрана модель, показавшая наилучшие результаты
- на основе выбранной модели была разработана программная реализация для решения задачи прогнозирования цен акций в определенный момент времени.

По результаты тестирования на основе показателей из индекса S&P500 можно сделать вывод, что использование разработанной программной реализации на основе модели Neural Prophet позволяет эффективно решать задачу прогнозирования финансовых временных рядов.

Список литературы

1. Кизбикенов, А. А. (2020). Основы эконометрики. — Текст: электронный // Алтайский государственный педагогический университет. — URL: <https://library.altspu.ru/dc/pdf/kizbikenov.pdf>
2. Теоретические основы нейронных сетей. — Текст: электронный // Мерионет. — URL: <https://wiki.merionet.ru/articles/teoreticheskie-osnovy-nejronnyh-setej>
3. Гафиятуллин, Г. Г. (2017). Нейронные сети: учебное пособие. — Текст: электронный // Казанский (Приволжский) федеральный университет. — URL: https://kpfu.ru/staff_files/F1493580427/NejronGafGal.pdf
4. Loginom. Моделирования ARIMA. — Текст: электронный // Loginom. — URL: <https://loginom.ru/blog/arima-example>
5. WallStreetMojo. Exponentially Weighted Moving Average (EWMA). — Текст: электронный // WallStreetMojo. — URL: <https://www.wallstreetmojo.com/ewma/>
6. Динамические модели. — Текст: электронный // Studfile.net. — URL: <https://studfile.net/preview/3320299/page:14/>
7. Быстрицкий, В. (2021). RNN, LSTM, GRU и их варианты. — Текст: электронный // Личный сайт Владимира Быстрицкого. — URL: http://vbystricky.ru/2021/05/rnn_lstm_gru_etc.html
8. Loginom. Метрики качества моделей. — Текст: электронный // Loginom. — URL: <https://loginom.ru/blog/quality-metrics>
9. Временные ряды. — Текст: электронный // Яндекс. Справочник. — URL: <https://education.yandex.ru/handbook/ml/article/vremennye-ryady>
10. Прогнозирование временных рядов с использованием Prophet от Facebook. — Текст: электронный // Facebook Research : [официальный сайт]. — URL: <https://facebook.github.io/prophet/>

11. NeuralProphet. — Текст: электронный // NeuralProphet. — URL: <https://neuralprophet.com/>
12. Груздев А. В. Г Прогнозирование временных рядов с помощью Facebook . — Текст: электронный // https://www.galaktika-dmk.com/upload/iblock/001/7ilgv6zyfufx33uxwoemzn19m1jvz3z2/45582-_978_5_93700_212_9_.pdf
13. УрФУ. Диссертация: "Исследование..." — Текст: электронный // Электронный архив УрФУ. — URL: <https://elar.urfu.ru/handle/10995/52487>

Приложение 1

Модель линейного роста Prophet:

```
# Импортирование необходимых библиотек
import pandas as pd
import matplotlib.pyplot as plt
from prophet import Prophet
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
from plotly import graph_objs as go

# Загрузка датасета
data = pd.read_csv('C:\\Users\\haida\\OneDrive\\Рабочий стол\\Study\\Диплом\\sp500.csv')
data

start_date = '2023-06-01'
end_date = '2024-05-31'
df1=pd.DataFrame()
df2=pd.DataFrame()
all_dates = pd.date_range(start=start_date, end=end_date)
df2['Date']= pd.to_datetime(all_dates)
df1['Date']=pd.to_datetime(data['Date'])
df1['Акции']=data.iloc[:,1:].sum(axis=1)
#df1=df1[:-60]
df = pd.merge(df2, df1, on='Date', how='left')
df['Date']=pd.to_datetime(df['Date'])
df['Акции'] = df['Акции'].fillna(method='ffill')

# инициализируем plotly
init_notebook_mode(connected = True)
```



```

# опишем функцию, которая будет визуализировать все колонки dataframe в виде l
ine plot
def plotly_df(df, title = ""):
    data = []

    for column in df.columns:
        if (column!= "Date"):
            trace = go.Scatter(
                x = df['Date'],
                y = df[column],
                mode = 'lines',
                name = column

            )
            data.append(trace)

    layout = dict(title = title)
    fig = dict(data = data, layout = layout)
    iplot(fig, show_link=False)

plotly_df(df, title = 'Показание акций')
[6]
predictions = 30

# приводим dataframe к нужному формату
df.columns = ['ds', 'y']
df.to_csv('SP500.csv')
# отрезаем из обучающей выборки последние 30 точек, чтобы измерить на них ка

```

чество

```
train_df = df[:-predictions]
```

```
train_df
```

```
start_date = '2023-06-01'
```

```
end_date = '2024-05-31'
```

```
# Генерируем все даты в указанном диапазоне
```

```
all_dates = pd.date_range(start=start_date, end=end_date)
```

```
# Фильтруем только субботы (5) и воскресенья (6)
```

```
weekends = all_dates[all_dates.weekday.isin([5, 6])]
```

```
# Создаем DataFrame с праздничными днями
```

```
holidays = pd.DataFrame({
```

```
    'holiday': 'weekend',
```

```
    'ds': weekends,
```

```
    'lower_window': 0,
```

```
    'upper_window': 1,
```

```
})
```

```
holidays
```

```
m = Prophet(holidays=holidays, changepoint_prior_scale=0.011)
```

```
m.add_country_holidays(country_name='US')
```

```
m.fit(train_df)
```

```
00:12:09 - cmdstanpy - INFO - Chain [1] start processing
```

```
00:12:09 - cmdstanpy - INFO - Chain [1] done processing
```

```
<prophet.forecaster.Prophet at 0x1e111037f10>
```

```
[13]
```

```
from prophet.plot import add_changepoints_to_plot
```

```
future = m.make_future_dataframe(periods=predictions)
```

```
forecast = m.predict(future)
```

```
fig = m.plot(forecast)
```

```
fig = m.plot_components(forecast)
```

```
[15]
```

```
forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail(30)
```

```
[16]
```

```
from prophet.diagnostics import cross_validation
```

```
from prophet.diagnostics import performance_metrics
```

```
df_cv = cross_validation(m, initial='300 days', horizon = '30 days')
```

```
df_p = performance_metrics(df_cv)
```

```
df_p
```

```
00:12:20 - cmdstanpy - INFO - Chain [1] start processing
```

```
00:12:20 - cmdstanpy - INFO - Chain [1] done processing
```

```
0%|          | 0/1 [00:00<?, ?it/s]
```

```
[17]
```

```
m = Prophet(changepoint_prior_scale=0.011, holidays=holidays)
```

```
m.add_country_holidays(country_name='US')
```

```
m.fit(train_df)
```

```
00:12:24 - cmdstanpy - INFO - Chain [1] start processing
```

```
00:12:24 - cmdstanpy - INFO - Chain [1] done processing
```

```
<prophet.forecaster.Prophet at 0x1e1134518e0>
```

```
[18]
```

```
from prophet.plot import plot_cross_validation_metric
```

```
fig = plot_cross_validation_metric(df_cv, metric='mape')
```

```

[19]
    cmp_df = forecast.set_index('ds')[['yhat', 'yhat_lower', 'yhat_upper']].join(df.set_index('ds'))
cmp_df

[43]
    import numpy as np
cmp_df['e'] = cmp_df['y'] - cmp_df['yhat']
cmp_df['p'] = 100*cmp_df['e']/cmp_df['y']
print(cmp_df[-predictions:][['e', 'p']])
print ('MAPE', np.mean(abs(cmp_df[-predictions:]['p'])))
print ('MAE', np.mean(abs(cmp_df[-predictions:]['e'])))

[44]
    def show_forecast(cmp_df, num_predictions, num_values):
# верхняя граница доверительного интервала прогноза
upper_bound = go.Scatter(
    name='Upper Bound',
    x=cmp_df.tail(num_predictions).index,
    y=cmp_df.tail(num_predictions)['yhat_upper'],
    mode='lines',
    marker=dict(color="#444444"),
    line=dict(width=0),
    fillcolor='rgba(68, 68, 68, 0.3)',
    fill='tonexty')

# прогноз
forecast = go.Scatter(
    name='Prediction',
    x=cmp_df.tail(num_predictions).index,

```

```
y=cmp_df.tail(num_predictions)['yhat'],  
mode='lines',  
line=dict(color='rgb(31, 119, 180)'),  
)
```

```
# нижняя граница доверительного интервала
```

```
lower_bound = go.Scatter(  
    name='Lower Bound',  
    x=cmp_df.tail(num_predictions).index,  
    y=cmp_df.tail(num_predictions)['yhat_lower'],  
    marker=dict(color="#447777"),  
    line=dict(width=0),  
    mode='lines')
```

```
# фактические значения
```

```
fact = go.Scatter(  
    name='Fact',  
    x=cmp_df.tail(num_values).index,  
    y=cmp_df.tail(num_values)['y'],  
    marker=dict(color="red"),  
    mode='lines',  
)
```

```
# последовательность рядов в данном случае важна из-за применения заливки
```

```
data = [lower_bound, upper_bound, forecast, fact]
```

```
layout = go.Layout(  
    yaxis=dict(title='Цена', gridcolor='rgba(200, 200, 200, 0.5)'),  
    xaxis=dict(title='Дата', gridcolor='rgba(200, 200, 200, 0.5)'),
```

```
title='Прогноз цены акций',  
showlegend=True,  
legend=dict(x=0, y=1, bgcolor='rgba(255, 255, 255, 0.5)'),  
plot_bgcolor='rgba(240, 240, 240, 0.9)'  
)
```

```
fig = go.Figure(data=data, layout=layout)  
iplot(fig, show_link=False)
```

Пример использования функции

предположим, что cmp_df - DataFrame с необходимыми столбцами, а predictions
- количество предсказаний

```
predictions = 30
```

Замените это на нужное значение

```
show_forecast(cmp_df, num_predictions=predictions, num_values=300)
```

Приложение 2

```
# Импортирование необходимых библиотек
import pandas as pd
import matplotlib.pyplot as plt
from prophet import Prophet
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
from plotly import graph_objs as go

[2]

# Загрузка датасета
data = pd.read_csv('C:\\Users\\haida\\OneDrive\\Рабочий стол\\Study\\Диплом\\sp500.
csv')
data

[3]

start_date = '2023-06-01'
end_date = '2024-05-31'
df1=pd.DataFrame()
df2=pd.DataFrame()
all_dates = pd.date_range(start=start_date, end=end_date)
df2['Date']= pd.to_datetime(all_dates)
df1['Date']=pd.to_datetime(data['Date'])
df1['Акции']=data.iloc[:,1:].sum(axis=1)
#df1=df1[:-60]
df = pd.merge(df2, df1, on='Date', how='left')
df['Date']=pd.to_datetime(df['Date'])
df['Акции'] = df['Акции'].fillna(method='ffill')
df

[4]
```

```

        # инициализируем plotly
init_notebook_mode(connected = True)

# опишем функцию, которая будет визуализировать все колонки dataframe в виде l
ine plot
def plotly_df(df, title = ""):
    data = []

    for column in df.columns:
        if (column!= "Date"):
            trace = go.Scatter(
                x = df['Date'],
                y = df[column],
                mode = 'lines',
                name = column

            )
            data.append(trace)

    layout = dict(title = title)
    fig = dict(data = data, layout = layout)
    iplot(fig, show_link=False)

plotly_df(df, title = 'Показание акций')
[5]
predictions = 30

# приводим dataframe к нужному формату

```



```
df.columns = ['ds', 'y']
```

```
# отрезаем из обучающей выборки последние 30 точек, чтобы измерить на них ка  
чество
```

```
train_df = df[:-predictions]
```

```
train_df['cap']=111000
```

```
train_df['floor']=83000
```

```
train_df
```

```
[6]
```

```
start_date = '2023-06-01'
```

```
end_date = '2024-05-31'
```

```
# Генерируем все даты в указанном диапазоне
```

```
all_dates = pd.date_range(start=start_date, end=end_date)
```

```
# Фильтруем только субботы (5) и воскресенья (6)
```

```
weekends = all_dates[all_dates.weekday.isin([5, 6])]
```

```
# Создаем DataFrame с праздничными днями
```

```
holidays = pd.DataFrame({
```

```
    'holiday': 'weekend',
```

```
    'ds': weekends,
```

```
    'lower_window': 0,
```

```
    'upper_window': 1,
```

```
})
```

```
holidays
```

```
[7]
```

```
m = Prophet(growth='logistic',holidays=holidays,changeoint_prior_scale=0.011)
m.add_country_holidays(country_name='US')
m.fit(train_df)
```

```
06:09:00 - cmdstanpy - INFO - Chain [1] start processing
06:09:00 - cmdstanpy - INFO - Chain [1] done processing
```

```
<prophet.forecaster.Prophet at 0x2c338548760>
[8]
from prophet.plot import add_changepoints_to_plot
future = m.make_future_dataframe(periods=predictions)
future['cap']=111000
future['floor']=83000
forecast = m.predict(future)

fig = m.plot(forecast)
```

```
[9]
fig = m.plot_components(forecast)

[10]
cmp_df = forecast.set_index('ds')[['yhat', 'yhat_lower', 'yhat_upper']].join(df.set_index('ds'))
cmp_df
[11]
from prophet.diagnostics import cross_validation
from prophet.diagnostics import performance_metrics
```

```

df_cv = cross_validation(m, initial='300 days', period='15 days', horizon = '30 days')
df_p = performance_metrics(df_cv)
df_p
06:34:15 - cmdstanpy - INFO - Chain [1] start processing
06:34:15 - cmdstanpy - INFO - Chain [1] done processing

0%|          | 0/1 [00:00<?, ?it/s]
[12]
from prophet.plot import plot_cross_validation_metric
fig = plot_cross_validation_metric(df_cv, metric='mape')

[13]
import numpy as np
cmp_df['e'] = cmp_df['y'] - cmp_df['yhat']
cmp_df['p'] = 100*cmp_df['e']/cmp_df['y']
print ('MAPE', np.mean(abs(cmp_df[-predictions:]['p'])))
print ('MAE', np.mean(abs(cmp_df[-predictions:]['e'])))
MAPE 1.2033418552866333
MAE 1300.9689080004225

cmp_df1=cmp_df.tail(30)
cmp_df['e'] = cmp_df['y'] - cmp_df['yhat']
cmp_df['p'] = 100*cmp_df['e']/cmp_df['y']
print ('MAPE', np.mean(abs(cmp_df[-predictions:]['p'])))
print ('MAE', np.mean(abs(cmp_df[-predictions:]['e'])))
[112]
def show_forecast(cmp_df, num_predictions, num_values):
# верхняя граница доверительного интервала прогноза
upper_bound = go.Scatter(
name='Upper Bound',

```

```

x=cmp_df.tail(num_predictions).index,
y=cmp_df.tail(num_predictions)['yhat_upper'],
mode='lines',
marker=dict(color="#444444"),
line=dict(width=0),
fillcolor='rgba(68, 68, 68, 0.3)',
fill='tonexty')

# прогноз
forecast = go.Scatter(
    name='Prediction',
    x=cmp_df.tail(num_predictions).index,
    y=cmp_df.tail(num_predictions)['yhat'],
    mode='lines',
    line=dict(color='rgb(31, 119, 180)'),
)

# нижняя граница доверительного интервала
lower_bound = go.Scatter(
    name='Lower Bound',
    x=cmp_df.tail(num_predictions).index,
    y=cmp_df.tail(num_predictions)['yhat_lower'],
    marker=dict(color="#447777"),
    line=dict(width=0),
    mode='lines')

# фактические значения
fact = go.Scatter(
    name='Fact',

```

```

x=cmp_df.tail(num_values).index,
y=cmp_df.tail(num_values)['y'],
marker=dict(color="red"),
mode='lines',
)

```

```

# последовательность рядов в данном случае важна из-за применения заливки
data = [lower_bound, upper_bound, forecast, fact]

```

```

layout = go.Layout(
    yaxis=dict(title='Цена', gridcolor='rgba(200, 200, 200, 0.5)'),
    xaxis=dict(title='Дата', gridcolor='rgba(200, 200, 200, 0.5)'),
    title='Прогноз цены акций',
    showlegend=True,
    legend=dict(x=0, y=1, bgcolor='rgba(255, 255, 255, 0.5)'),
    plot_bgcolor='rgba(240, 240, 240, 0.9)'
)

```

```

fig = go.Figure(data=data, layout=layout)
iplot(fig, show_link=False)

```

Пример использования функции

предположим, что cmp_df - DataFrame с необходимыми столбцами, а predictions
- количество предсказаний

```
predictions = 30
```

Замените это на нужное значение

```
show_forecast(cmp_df, num_predictions=predictions, num_values=200)
```

Приложение 3

```
import sys
import pandas as pd
from predict import ProphetModel

if __name__ == "__main__":
    if len(sys.argv) != 3:
        print("Usage: python predict_script.py <data_path> <period>")
        sys.exit(1)

    data_path = sys.argv[1]
    period = int(sys.argv[2])

    # Load data
    df = pd.read_csv(data_path)
    # Initialize ProphetModel with default model path
    model = ProphetModel()

    # Make prediction
    forecast = model.predict(df, period)
    forecast = forecast.join(df['y'], how='left')
    forecast.to_csv('result_of_forecast.csv')
    # Print forecast
    print(forecast)

import pandas as pd
from prophet.serialize import model_to_json, model_from_json
```

```

class ProphetModel:
    def __init__(self, model_path='final_model.json'):
        self.model_path = model_path
        self.model = self.load_model()

    def load_model(self):
        with open('final_model.json', 'r') as fin:
            model = model_from_json(fin.read()) # Load model
        return model

    def predict(self, df, period):
        # Create future dataframe
        future = self.model.make_future_dataframe(periods=period)

        # Make forecast
        forecast = self.model.predict(future)

        return forecast[['ds', 'yhat']].tail(period)

```

Приложение 4

```
import pandas as pd

from neuralprophet import NeuralProphet
from sklearn.metrics import mean_absolute_error, mean_absolute_percentage_error
import numpy as np
import matplotlib.pyplot as plt
df = pd.read_csv('SP500.csv')
print(df.head())
prediction=30
df_train = df.iloc[:-prediction]
df_train = df_train.drop('Unnamed: 0', axis=1)
df_test = df.iloc[-prediction:]
nprophet_model = NeuralProphet()
metrics = nprophet_model.fit(df_train, freq="D")
future_df = nprophet_model.make_future_dataframe(df_train,
                                                periods =10,
                                                n_historic_predictions=len(df_train))
preds_df_2 = nprophet_model.predict(future_df)

preds_df_2.to_csv('Neural.csv', index=False)
print(f"NeuralProphet MAE:\t{mean_absolute_error(df_test['y'], preds_df_2.iloc[-prediction:]['yhat1']):.4f}")
MAPE=mean_absolute_percentage_error(df_test['y'], preds_df_2.iloc[-prediction:]['yhat1'])*100
print(f"NeuralProphet MAPE:\t{MAPE:.4f}")
```