

Санкт–Петербургский государственный университет

КНЯЗЕВ Никита Андреевич

Выпускная квалификационная работа

***Прогнозирование состояния сложных сетевых систем с
использованием методов машинного обучения***

Уровень образования: магистратура

Направление 03.04.01 «Прикладные математика и физика»

Основная образовательная программа ВМ.5521.2021 «Математические и
информационные технологии»

Научный руководитель:

доцент, кафедра теории систем управления элек-
трофизической аппаратурой,

к. ф.-м. н. Козынченко Владимир Александрович

Санкт-Петербург

2023 г.

Содержание

| | |
|---|----|
| Введение | 4 |
| Обзор литературы | 6 |
| Постановка задачи | 8 |
| Глава 1. Анализ временных рядов | 9 |
| 1.1. Основные определения | 9 |
| 1.2. Прогнозирование временных рядов на сетях | 11 |
| Глава 2. Описание алгоритмов | 14 |
| 2.1. Алгоритмы прогнозирования | 14 |
| 2.1.1 Авторегрессионная модель | 15 |
| 2.1.2 Векторная авторегрессионная модель | 17 |
| 2.1.3 Методы градиентного бустинга | 18 |
| 2.2. Алгоритмы сглаживания | 25 |
| 2.3. Алгоритм валидации | 30 |
| Глава 3. Анализ данных | 33 |
| 3.1. Abilene | 33 |
| 3.2. Totem | 39 |
| 3.3. PeMSD7 | 44 |
| 3.4. Вывод | 49 |
| Глава 4. Вычислительные эксперименты | 50 |
| 4.1. Подготовка среды | 50 |
| 4.2. Тестирование и выбор основных методов | 51 |
| 4.3. Результаты валидации | 53 |
| 4.3.1 Одномерные модели | 53 |
| 4.3.2 Многомерные модели | 55 |
| 4.4. Анализ результатов | 57 |
| Заключение | 59 |
| Список литературы | 61 |
| Приложение А. Обзор литературы | 64 |

| | |
|---|-----------|
| Приложение Б. Результаты валидации одномерных моделей для | |
| трафика по узлам | 68 |
| Приложение В. Результаты валидации многомерных моделей для | |
| трафика по узлам | 74 |
| Приложение Г. Визуализация результатов прогнозирования | 80 |

Введение

С появлением и быстрым развитием компьютерных технологий и Интернета к сети подключается все большее количество устройств, что приводит к увеличению ее масштабов и сложности, а запросы пользователей к качеству связи только повышаются. Разумно распределяя сетевые ресурсы, можно оптимизировать маршрутизацию, тем самым уменьшить задержку связи, предотвратить перегрузку сети и обеспечить качество услуг. Однако для диагностики сети, обнаружения аномалий и эффективного использования имеющихся ресурсов необходимо точное прогнозирование ключевых показателей сети, таких как трафик, задержки, вызовы и т. д. Таким образом, прогнозирование сетевого трафика является основой для улучшения качества обслуживания пользователей.

С телекоммуникационной точки зрения сетевой трафик представляет собой матрицу, описывающую объем потока данных между всеми парами узлов сети в данный момент времени. Задачу прогнозирования сетевого трафика можно сформулировать как предсказание данной матрицы в определенный момент времени в будущем на основе исторических данных. Для решения поставленной задачи используют методы, которые можно разделить на две группы: линейные и нелинейные. Линейные методы, такие как авторегрессионные модели [19, 23], моделируют характеристики временной последовательности трафика на основе математической статистики. К нелинейным относятся модели прогнозирования, основанные на вейвлет-анализе [7, 2], байесовских сетях [21], нейронных сетях [20, 5, 6] и так далее. Поскольку одна линейная или нелинейная модель не может точно описать динамику сетевого трафика, то для решения подобных задач в последнее время все чаще стали использовать комбинированные методы [12], основанные на нейронных сетях, способных моделировать пространственные и временные признаки. Однако и у данных методов есть свои недостатки. Например, в некоторых случаях сначала извлекаются пространственные характеристики с использованием графовой нейронной сети (*GNN* [24, 11]), а затем захватываются временные признаки. Методы на основе *GNN* создают статическую матрицу смежности для моделирования топологии сети, где каждый узел передает

сообщения только своим непосредственным соседям. При этом не учитывается тот факт, что пространственные зависимости сети динамичны, то есть важность различных узлов меняется со временем, что делает структуру графа более сложной. Такие методы могут сильно ограничивать возможности моделирования сложного сетевого трафика.

Для решения задачи прогнозирования сетевого трафика мы предлагаем использовать ряд линейных и нелинейных методов, дополнив их возможностью обрабатывать пространственную информацию. Также в работе проведен анализ временных и пространственных зависимостей данных, определенных на графах и сетях, и рассмотрены различные способы фильтрации для уменьшения влияния шума на модели.

Областью исследования данной работы являются методы машинного обучения. Предмет исследования – данные с сетевой структурой. В данной работе предстоит решить такие проблемы как: выбор подходящих методов прогнозирования; поиск открытых баз данных; разработка алгоритмов прогнозирования; тестирование и анализ полученных результатов.

В работе приведен обзор литературы по изучаемой теме и сформулированы цель и задачи исследования. В первой главе представлены некоторые теоретические сведения по теории временных рядов и их прогнозированию. Вторая глава посвящена описанию алгоритмов, которые будут исследованы. В третьей главе проведен корреляционный анализ рассматриваемых наборов данных. В четвертой главе представлены результаты вычислительных экспериментов. В заключении подведены итоги проведенной работы.

Обзор литературы

Существующие методы решения поставленной задачи.

При проведении исследования были изучены существующие решения задачи прогнозирования временных рядов, определенных на сетях. Некоторые из них используют классические статистические методы ([19, 23]), но в настоящее время наблюдается тенденция к использованию алгоритмов машинного обучения ([20, 5, 6]), среди которых выделяются графовые сверточные сети ([24, 12, 11]). Главной проблемой современных решений является сложность модели: обычно они имеют множество слоев с огромным количеством связей, требуют много времени и вычислительных ресурсов для обучения, а настройка параметров может быть довольно сложной. Будучи моделями черного ящика, им также не хватает интерпретируемости и объяснимости по сравнению со статистическими методами.

Также важно отметить, что рассмотренные статьи не содержат никакой информации о валидации или используют только тривиальные процедуры тестирования: была найдена только одна статья, определяющая алгоритм валидации для поиска параметров и оценки результатов [17].

Более двух третей рассмотренных работ связаны с дорожным движением, в то время как прогнозирование сетевого трафика остается недостаточно изученным, несмотря на большой интерес и спрос со стороны отрасли. Это можно объяснить большей сложностью временных рядов сетевого трафика: их динамика гораздо более стохастическая и зашумленная, а связи между узлами могут быть прерывистыми. Однако прогнозирование сетевого трафика имеет более широкую область применения, включая оптимизацию сети, контроль и обнаружение неисправностей, в то время как предсказание дорожного движения обычно ограничивается решением транспортных задач (например, поиском оптимального пути от узла А к узлу Б), так как мы не можем ничего изменить или настроить в дорожной сети.

В таблице 8 (Приложение А) представлен полный обзор литературы по изучаемой теме.

Обзор методов онлайн обучения и адаптации.

Учитывая, что сетевой трафик может переходить между различными динамическими режимами, важно убедиться, что модель прогнозирования способна адаптироваться к внезапным изменениям. Другими словами, модель прогнозирования должна быть дополнена алгоритмом онлайн-адаптации. Поэтому мы решили сделать обзор некоторых популярных методов онлайн-обучения и адаптации (Таблица 9, Приложение А).

Некоторые работы (в частности, [23, 4]) используют онлайн-градиентный спуск в качестве подхода к онлайн-обучению. Градиентный спуск – это итеративные алгоритмы оптимизации первого порядка для минимизации целевой функции, активно используемые в машинном обучении. Онлайн-градиентный спуск – его вариант, основанный на пакетном или стохастическом градиентном спуске, при этом пакетный вариант имеет более стабильную сходимость. Но у градиентного спуска есть существенный недостаток: он может сходиться к локальному минимуму и седловым точкам (глобальный минимум гарантированно будет найден только в том случае, когда целевая функция выпуклая).

Еще одним методом адаптации является метод скользящего окна или его модификации [16]. Метод скользящего окна – это алгоритм преобразования, основанный на сдвиге интервала со значениями, который позволяет генерировать подвыборки временного ряда для обучения и тестирования модели прогнозирования. Процесс работы данного метода заключается в следующем: окно смещается по последовательности на единицу наблюдения, где каждая позиция образует новый пример.

В результате мы будем использовать метод скользящего окна для онлайн-адаптации моделей, что подробно описывается в разделе 2.3.

Постановка задачи

Основной целью работы является поиск и разработка эффективных алгоритмов и подходов для прогнозирования временных рядов сетевого трафика в реальном времени на основе исторических наблюдений за атрибутами сети. Данная работа ориентирована только на краткосрочное прогнозирование, подразумевая, что горизонт предсказания намного меньше исторической записи, используемой для обучения, и сравним с периодом типичных колебаний в системе. Однако, в отличие от большинства работ, посвященных краткосрочному прогнозированию временных рядов, мы не ограничиваемся одно- или двухэтапным прогнозированием, а пытаемся исследовать предсказательную способность наших моделей на десятки и даже сотни временных шагов. Результаты исследования могут быть применены к целому ряду задач, от контроля и мониторинга телекоммуникационных сетей до оптимизации сети с обратной связью.

Основываясь на данной цели, были сформулированы основные задачи:

1. провести анализ существующих решений поставленной проблемы;
2. найти подходящие открытые наборы данных, содержащих временные ряды и имеющих графовую структуру;
3. подобрать подходящие алгоритмы для фильтрации, сглаживания, прогнозирования данных и валидации будущих моделей;
4. выполнить настройку параметров и гипер-параметров моделей;
5. обучить модели и выполнить их валидацию;
6. провести анализ полученных результатов.

Глава 1. Анализ временных рядов

1.1 Основные определения

Под *временным рядом* будем понимать множество наблюдений X_t , собранных в дискретное время $t \in \mathbb{N} \cup \{0\}$. Дискретное время t связано с некоторым физическим временем, тогда как шаг физического времени между любыми двумя соседними дискретными моментами времени t и $t+1$ всегда постоянен. Реальные измерения (наблюдения) редко удовлетворяют последнему условию, поскольку они часто собираются с нерегулярными временными шагами. В результате обычно можно найти наблюдения, подобные показанным на верхнем графике рисунка 1. Затем требуется процедура предварительной обработки, чтобы превратить эти наблюдения во временные ряды, соответствующие вышеупомянутому определению. Нижний график на том же рисунке иллюстрирует возможный способ сделать это. Здесь мы определяем «ожидаемый временной шаг», а затем генерируем новые значения в дискретные моменты времени (штрихи на оси времени), выбранные с ожидаемым временным шагом путем интерполяции доступных наблюдений. В данном конкретном примере для этой цели использовалась кусочно-постоянная интерполяция (или, что то же самое, интерполяция ближайшего соседа) в сочетании с условием, утверждающим, что если расстояние по времени от X_t до ближайшего наблюдения больше, чем удвоенный ожидаемый временной шаг, то мы присваиваем *None*. Ясно, что эту процедуру можно легко обобщить на произвольные схемы интерполяции, такие как интерполяция Лагранжа высокого порядка или сплайны, в зависимости от практической задачи.

Временные ряды, рассматриваемые в данной работе, происходят из дискретных процессов, определенных в сетях. Типичным примером является сквозная коммуникационная сеть, в которой такой дискретный процесс связан с потоками трафика, т. е. пакетами, отправляемыми от исходного узла к узлу назначения. Конкретным дискретным событием в этом случае является, например, отправка узлом A N байтов узлу B в заданное время. На микроскопическом уровне такие дискретные события можно рассматривать как случайные события, происходящие в случайные моменты времени, и,

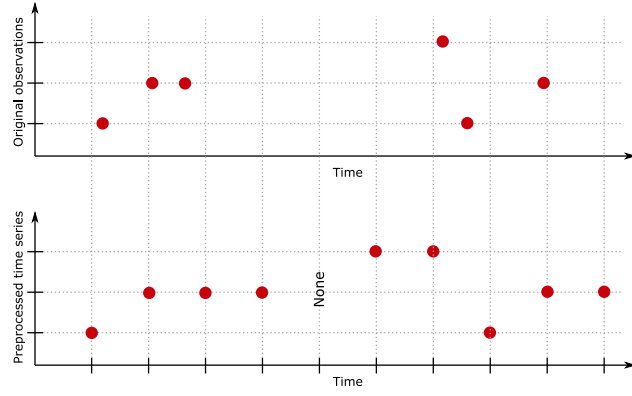


Рис. 1: Иллюстрация типичной процедуры предварительной обработки, необходимой для реальных наблюдений. Верхний график показывает исходные наблюдения. На нижнем графике показаны скорректированные наблюдения, дающие временной ряд с постоянным временным шагом.

таким образом, их нельзя смоделировать с помощью временных рядов, подразумевающих регулярные временные шаги. Вместо этого мы должны агрегировать такие события за конечные интервалы времени. Например, подсчет количества событий, происходящих в течение заданного интервала времени, приведет к гистограммам, а измерение объема трафика в байтах, проходящих через данный узел в течение заданного интервала времени, приведет к скорости потока трафика. Если выбранный временной интервал не слишком мал, чтобы допустить случайность, и не слишком велик, чтобы размыть любую нетривиальную динамику, эта процедура создает макроскопическое, хотя и динамическое представление дискретного процесса, которое обычно моделируется временными рядами в сетях.

Временные ряды в сетях можно определить несколькими способами. Рассмотрим сеть $G = (V, E)$, где V – множество узлов, а E – множество ребер. Мы можем связать уникальный временной ряд с:

1. каждым узлом сети $v \in V$ (например, $X_{v,t}$ может обозначать трафик, проходящий через узел v в момент времени t);
2. каждым ребром сети $e \in E$ (например, $X_{e,t}$ может обозначать трафик, проходящий через ребро e в момент времени t);
3. каждой парой узлов сети $(v_1, v_2) \in V \times V$ (например, $X_{v_1, v_2, t}$ может обозначать поток трафика из источника v_1 в пункт назначения v_2 в момент времени t).

Это приводит к *многомерным временным рядам* \mathbf{X}_t (например, $\mathbf{X}_t = [\mathbf{X}_{v_1,t}, \dots, \mathbf{X}_{v_N,t}]^T$ для временных рядов, определенных только на узлах), отличающихся собственным набором методов анализа и прогнозирования по сравнению с одномерными [13]. Таким образом, под временным рядом на сети мы понимаем многомерный временной ряд \mathbf{X}_t и граф $G = (V, E)$.

1.2 Прогнозирование временных рядов на сетях

Прежде всего, прогнозирование временных рядов подразумевает построение такой функции $f(\cdot)$, которая может предсказать значение \mathbf{X}_{t+1} на основе исторических наблюдений того же временного ряда:

$$\mathbf{X}_{t+1} = f(\mathbf{X}_t, \mathbf{X}_{t-1}, \dots, \mathbf{X}_{t-p}; \mathbf{a}),$$

где $t, t-1, \dots, t-p$ называются *лагами*, а $\mathbf{a} \in \mathbb{R}^n$ – вектором параметров функции $f(\cdot)$.

Различные методы прогнозирования временных рядов различаются тем, как они определяют функцию $f(\cdot)$ и какой алгоритм используют для нахождения оптимальных значений параметров \mathbf{a} .

Базовые подходы предполагают, что одномерные компоненты многомерного временного ряда \mathbf{X}_{t+1} на самом деле несвязаны (т. е. не зависят друг от друга) и, таким образом, могут моделироваться и прогнозироваться по отдельности. Одной из популярных моделей такого одномерного прогнозирования является *авторегрессионная (AR) модель*, в которой функция $f(\cdot)$ представляет собой линейную комбинацию лагов:

$$X_{t+1} = \sum_{i=1}^N a_i X_{t-i} + \text{iid шум}, \quad t = N, \dots, T,$$

где T – максимальный временной горизонт, а N – количество лагов, которые мы хотим использовать для прогнозирования.

Более сложные подходы направлены на моделирование взаимозависимости между различными компонентами многомерного временного ряда \mathbf{X}_{t+1} . Подчеркнем, что подавляющее большинство существующих методов

не учитывает структуру (граф), лежащую в основе многомерного временного ряда, в то время как очевидно, что сетевые связи определяют пространственную зависимость между компонентами временного ряда.

Важно отметить, что любой метод прогнозирования предполагает, что наблюдаемый временной ряд может быть разложен на *детерминированные* и *стохастические* составляющие. Детерминированные переменные описывают плавную динамику, которую можно связать с *трендом* (долгосрочный рост или спад) и *сезонностью* (систематические периодические колебания). Обычно они хорошо предсказуемы, например, с помощью методов экстраполяции или подходов, основанных на моделировании. Стохастическая составляющая описывает неравномерную, зашумленную динамику, связанную со *случайными процессами* или *шумом*. Ее можно предсказать только в вероятностном смысле, подразумевая, что X_t оценивается как случайная величина со своим собственным распределением. Вопросы о том, как следует сочетать детерминистические и стохастические компоненты и как следует вводить шумовые составляющие, являются фундаментальными для всей области прогнозирования временных рядов.

Проблема прогнозирования временных рядов состоит в том, чтобы предсказать будущие значения на основе исторических наблюдений за одним и тем же временным рядом. Количество временных шагов, для которых должен быть сделан прогноз, называется *горизонтом прогноза*. В зависимости от горизонта задачу можно рассматривать как задачу *краткосрочного прогнозирования* или *долгосрочного*. Иллюстрация первого случая показана на рисунке 2. Отметим, что горизонт прогнозирования значительно меньше периода слабых колебаний в этой системе. Именно это условие позволяет назвать прогнозирование краткосрочным. Также важно отличать *одношаговый прогноз* (см. рисунок 2). В то время как одношаговое прогнозирование активно используется для экономических и маркетинговых задач (например, прогнозирование продаж), оно вряд ли будет полезно для задач управления или самовосстановления, возникающих в автономных телекоммуникационных сетях. Таким образом, мы сосредоточимся на многоэтапном прогнозировании с относительно длинным временным горизонтом, но все же достаточно коротким, чтобы его можно было классифицировать как краткосрочное

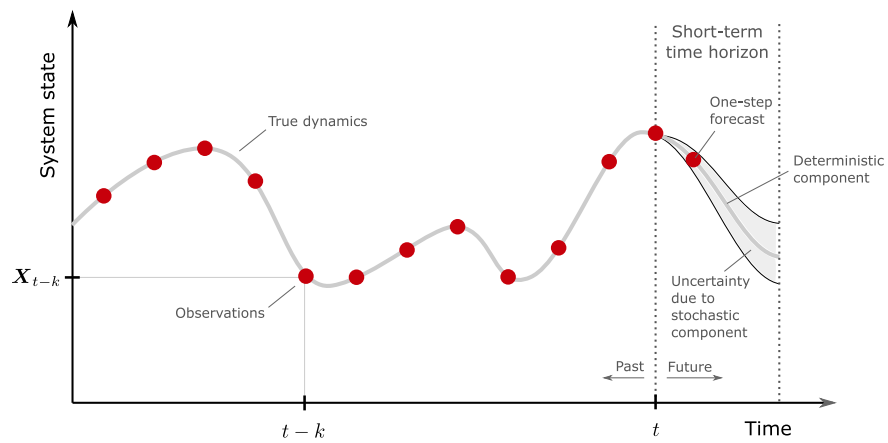


Рис. 2: Иллюстрация одношагового прогнозирования и краткосрочного прогнозирования с расчетной неопределенностью на основе исторических наблюдений за временными рядами.

прогнозирование. На наш взгляд, это направление недостаточно освещено в литературе, несмотря на его явную актуальность для приложений телекоммуникационных сетей.

Глава 2. Описание алгоритмов

2.1 Алгоритмы прогнозирования

В этом исследовании мы ограничимся двумя группами моделей, используемых для прогнозирования временных рядов: статистическими и методами, основанными на деревьях решений.

Статистические модели, такие как *ARIMA*, векторная авторегрессия, экспоненциальное сглаживание и т. д., основаны на объемной теоретической базе и успешно используются в ряде областей [3, 13]. Например, они являются стандартом в финансовом анализе, эконометрике и продажах, представленных зашумленными и заведомо сложными временными рядами. В этих областях прогнозирование является частью принятия решений, подразумевая, что модель прогнозирования должна каким-то образом предоставлять информацию для вероятностной оценки риска. Статистические модели очень хорошо подходят для этой цели, что объясняет их популярность.

Вторая группа методов не обладает такими свойствами. Тем не менее, все больше исследований показывают, что методы, основанные на деревьях решений, весьма эффективны при решении задач прогнозирования временных рядов, продемонстрированных на больших наборах данных различных типов. Например, 4 из 5 лучших моделей в конкурсе «M5 Competition» были основаны на LightGBM [15]. Следовательно, имеет смысл проверить их эффективность для временных рядов трафика, редко представленных в наборах данных общего типа.

2.1.1 Авторегрессионная модель

Авторегрессионная модель $AR(p)$ используется для прогнозирования одномерных временных рядов и предполагает, что X_t можно оценить как линейную комбинацию значений временных рядов при предыдущих p лагах:

$$X_t = c + \sum_{i=1}^p \varphi_i X_{t-i} + \epsilon_t,$$

где

$\mathbf{a} = [c, \varphi_1, \dots, \varphi_p]^T$ – параметры модели,

c – пересечение или смещение;

ϵ_t – белый шум.

Часто на прогнозируемый временной ряд X_t влияют другие переменные, к которым у нас может быть доступ (например, управляющие сигналы или поведение пользователя). Эти дополнительные внешние переменные называются *экзогенными* переменными, тогда как X_t называется *эндогенной* переменной. Если дополнить $AR(p)$ экзогенными переменными, мы получим *авторегрессионную экзогенную модель* $AR(p, m)$:

$$X_t = c + \sum_{i=1}^p \varphi_i X_{t-i} + \sum_{i=1}^k \sum_{j=1}^m \mu_{ij} Y_{i,t-j} + \epsilon_t,$$

где

$Y_{i,t}$ – значение i -го экзогенного входа в момент времени t ;

k – количество экзогенных переменных;

m – количество лагов для экзогенных переменных;

$\mathbf{a} = [c, \varphi_1, \dots, \varphi_p, \mu_{11}, \dots, \mu_{km}]^T$ – параметры модели.

Для реальных временных рядов характерна некоторая сезонность с одним или несколькими периодами, поэтому необходимо иметь возможность включить эту часть информации в модель. Сезонные авторегрессионные модели можно получить двумя альтернативными способами: либо с использованием сезонного порядка, либо с использованием сезонных фиктивных переменных. Сейчас мы рассмотрим первый вариант, второй будет представлен далее по тексту. Сезонный порядок l определяет, на сколько сезонов мы должны оглянуться назад за период T :

$$X_t = c + \sum_{i=1}^p \varphi_i X_{t-i} + \sum_{i=1}^l \nu_i X_{t-iT} + \epsilon_t,$$

где вектор параметров модели принимает вид $\mathbf{a} = [c, \varphi_1, \dots, \varphi_p, \nu_1, \dots, \nu_l]^T$.

Стабильность авторегрессионной модели зависит от формы аппроксимируемого временного ряда. В частности, процессы *AR/VAR* являются стационарными тогда и только тогда, когда корни авторегрессионного многочлена лежат вне единичного круга. Обычно алгоритмы оценки не накладывают это условие во время поиска параметров, что может привести к нестабильности модели и впоследствии к расходящемуся прогнозу. Пример такой нестабильности можно увидеть на рисунке 3. Чтобы не получать плохие значения оценивающих метрик, мы заменим такие нестабильные модели наивным прогнозом, т. е. средним значением (рис. 4).

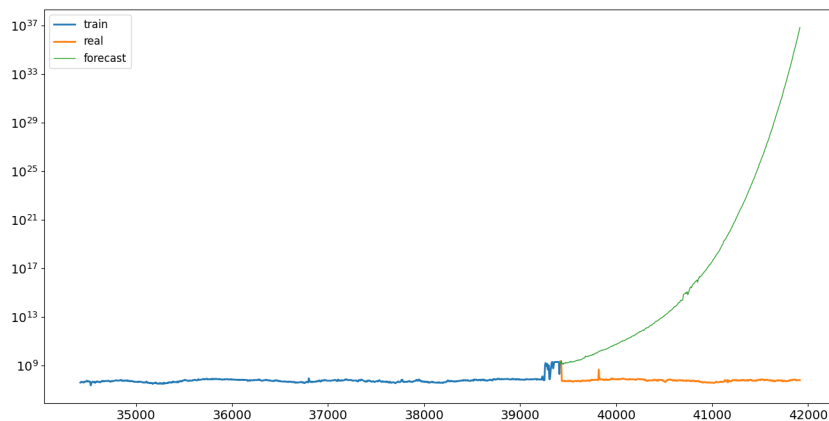


Рис. 3: Пример нестабильного поведения AR модели.

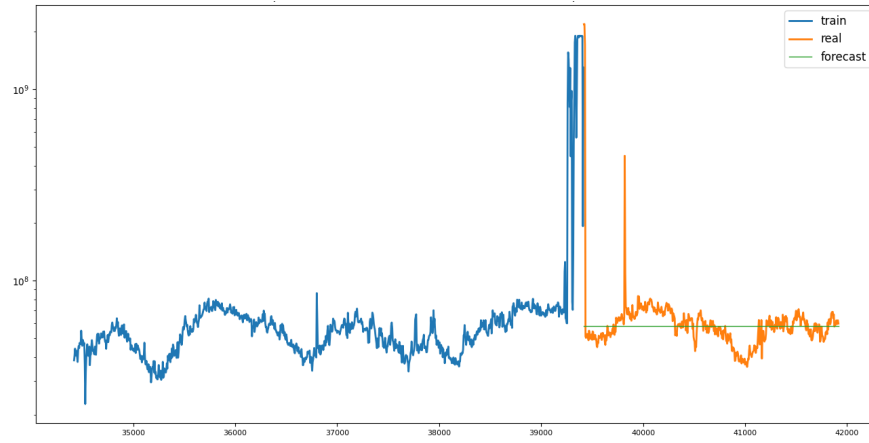


Рис. 4: Наивный метод решения неустойчивости авторегрессионной модели.

2.1.2 Векторная авторегрессионная модель

Векторная авторегрессионная модель $VAR(p)$ – это расширение авторегрессионной модели на случай многомерных временных рядов:

$$\mathbf{X}_t = \mathbf{c} + \sum_{i=1}^p A_i \mathbf{X}_{t-i} + \epsilon_t.$$

где параметры модели \mathbf{a} теперь состоят из вектора пересечения \mathbf{c} и матриц A_1, \dots, A_p .

Вектор шума ϵ_t должен удовлетворять следующим условиям:

1. $E(\epsilon_t) = 0$ (каждая переменная шума имеет нулевое среднее значение);
2. $E(\epsilon_t \epsilon_t') = \Omega \succcurlyeq 0$ (одновременная ковариационная матрица шума представляет собой $(k \times k)$ положительно-полуопределенную матрицу, Ω);
3. $E(\epsilon_t \epsilon_{t-i}') = 0$ для любого ненулевого i (нет корреляции во времени).

Сезонность также может быть введена для моделей VAR . Ее проще реализовать с помощью *фиктивных сезонных переменных (dummy variables)*.

Сезонные переменные. Предполагая s периодов, сезонность можно наложить на модель VAR , добавив *сезонные фиктивные переменные* n_{it} :

$$\mathbf{X}_t = \mathbf{c} + \sum_{i=1}^s n_{it} v_i + \sum_{i=1}^p A_i \mathbf{X}_{t-i} + \epsilon_t.$$

где $n_{it} = 1$, если t принадлежит i -му сезону, или нулю в противном случае.

Это предполагает, что $\sum_{i=1}^s n_{it} = 1$ для любого t . Таким образом, параметры модели \mathbf{a} состоят из \mathbf{c} , v_1, \dots, v_s и A_1, \dots, A_p , где \mathbf{c} неявно включен в v_i . Более общую форму сезонности можно получить, если матрицы A_i будут зависеть от времени года.

Количество используемых временных лагов авторегрессионных моделей (AR/VAR) можно настроить методом `select_order` из библиотеки `statsmodels`, который использует для этого информационные критерии. Мы сосредоточились на информационном критерии Акаике (AIC) – оценщике ошибки прогнозирования и, следовательно, относительного качества статистических моделей для заданного набора данных.

2.1.3 Методы градиентного бустинга

Вообще говоря, *бустинг* – это ансамблевый алгоритм, который строит сильную модель, используя набор слабых. Известные алгоритмы бустинга, такие как XGBoost, LightGBM и CatBoost, используют деревья решений f_k в качестве слабых моделей, поэтому ансамбль, который генерирует прогнозы для X_t , принимает форму:

$$\hat{X}_t^{(K)} = \sum_{k=1}^K f_k(X_{t-1}, \dots, X_{t-i}), \quad f_k \in \mathcal{F},$$

где

$\hat{X}_t^{(K)}$ – значение временного ряда, прогнозируемое ансамблем с K слабыми моделями;

K – количество деревьев;

\mathcal{F} – множество всех возможных деревьев решений;

f_k – функция, ассоциированная с деревом решений в функциональном пространстве \mathcal{F} .

Под деревом решений мы понимаем дерево, в котором каждый внутренний узел обозначает проверку атрибута, каждая ветвь представляет результат проверки, а каждый конечный (листовой) узел содержит оценку. Целевая функция, которую необходимо оптимизировать, имеет вид:

$$obj = \sum_{t=0}^T L(X_t, \hat{X}_t^{(K)}) + \sum_{k=1}^K \omega(f_k),$$

где

L – функция потерь при обучении;

$\omega(f_k)$ – сложность дерева f_k .

Изучение древовидной структуры намного сложнее, чем большинство задач оптимизации машинного обучения, где можно вычислить градиент, например, через обратное распространение ошибки. Задача оптимизации для ансамбля деревьев, напротив, является комбинаторной и, следовательно, трудноразрешимой. Уловка, которую использует градиентный бустинг, заключается в том, что он не пытается изучить все f_k сразу, а изучает их итеративно. Сначала из обучающих данных строится и фиксируется исходная слабая модель f_1 . Затем вторая модель f_2 обучается на ошибках первой модели, третья модель f_3 обучается на ошибках f_2 и так далее [10]:

$$\begin{aligned}\hat{X}_t^{(0)} &= 0 \\ \hat{X}_t^{(1)} &= f_1 = \hat{X}_t^{(0)} + f_1 \\ \hat{X}_t^{(2)} &= f_1 + f_2 = \hat{X}_t^{(1)} + f_2 \\ &\dots \\ \hat{X}_t^{(k)} &= \sum_{j=1}^k f_j = \hat{X}_t^{(k-1)} + f_k\end{aligned}$$

На итерации k мы хотим найти такую слабую модель f_k , чтобы минимизировалась следующая целевая функция:

$$\begin{aligned} obj^{(k)} &= \sum_{t=0}^T L(X_t, \hat{X}_t^{(k)}) + \sum_{k=1}^K \omega(f_k) = \\ &= \sum_{t=0}^T L(X_t, \hat{X}_t^{(k)}) + \omega(f_K) + constant \rightarrow \min \end{aligned}$$

В задачах регрессии, включая краткосрочное прогнозирование, функция потерь обычно представляет собой среднеквадратичную ошибку, что превращает целевую функцию в

$$\begin{aligned} obj^{(k)} &= \sum_{t=0}^T \left(X_t - \hat{X}_t^{(k)} \right)^2 + \omega(f_k) + constant = \\ &= \sum_{t=0}^T \left(X_t - (\hat{X}_t^{(k-1)} + f_k) \right)^2 + \omega(f_k) + constant = \\ &= \sum_{t=0}^T \left[2(\hat{X}_t^{(k-1)} - X_t)f_k + f_k^2 \right] + \omega(f_k) + constant \end{aligned}$$

где константа поглощает фиксированные в данной оптимизационной задаче переменные.

В более общем случае мы берем разложение Тейлора функции потерь до второго порядка и получаем:

$$\begin{aligned} obj^{(k)} &= \sum_{t=0}^T \left[L(X_t, \hat{X}_t^{(k-1)}) + g_t f_k + \frac{1}{2} h_t f_k^2 \right] + \omega(f_k) + constant \\ g_t &= \partial_{\hat{X}_t^{(k-1)}} L(X_t, \hat{X}_t^{(k-1)}) \\ h_t &= \partial_{\hat{X}_t^{(k-1)}}^2 L(X_t, \hat{X}_t^{(k-1)}) \end{aligned}$$

Таким образом, окончательный вариант задачи имеет следующий вид:

$$obj^{(k)} = \sum_{t=0}^T [g_t f_k + \frac{1}{2} h_t f_k^2] + \omega(f_k). \quad (1)$$

Наша цель – минимизировать (1) по f_k , т. е. это оптимизационная задача, определенная на функциональном пространстве. Для ее решения введем следующее определение f_k :

$$f_k(x) = \omega_{q(x)},$$

где

$$\omega \in \mathbb{R}^L,$$

q присваивает индекс листьев $1, \dots, L$ для x ,

L – общее количество листьев в дереве.

Определим сложность дерева как штраф, сочетающий Тихоновскую регуляризацию и количество листьев L :

$$\omega(f_k) = \gamma L + \frac{1}{2} \lambda \sum_{j=1}^L \omega_j^2.$$

Это позволяет получить аналитическое решение для $\omega_1, \dots, \omega_L$:

$$\begin{aligned} obj^{(k)} &= \sum_{t=0}^T \left[g_t \omega_{q(x_t)} + \frac{1}{2} h_t \omega_{q(x_t)}^2 \right] + \gamma L + \frac{1}{2} \lambda \sum_{j=1}^L \omega_j^2 \\ &= \sum_{j=1}^L \left[\left(\sum_{i \in I_j} g_i \right) \omega_j + \frac{1}{2} \left(\sum_{i \in I_j} h_i + \lambda \right) \omega_j^2 \right] + \gamma L \\ &= \sum_{j=1}^L \left[G_j \omega_j + \frac{1}{2} (H_j + \lambda) \omega_j^2 \right] + \gamma L, \end{aligned}$$

где

I_j – набор индексов $\{x_i\}$, для которых $q(x_i) = j$,

$$G_j = \sum_{i \in I_j} g_i,$$

$$H_j = \sum_{i \in I_j} h_i.$$

Решение для нулей частных производных дает:

$$\begin{aligned}\omega_j^* &= -\frac{G_j}{H_j + \lambda}, \\ obj^* &= \frac{1}{2} \sum_{j=1}^L \frac{G_j^2}{H_j + \lambda} + \lambda L.\end{aligned}$$

Заметим, что это решение сформулировано для фиксированной древовидной структуры $q(x)$. Таким образом, obj^* измеряет качество дерева со структурой $q(x)$. В идеале мы могли бы перебрать все возможные комбинации $q(x)$ и выбрать наилучшую, но это явно невозможно. Общая идея состоит в том, чтобы просто увеличить древовидную структуру, используемую в $k - 1$, либо по горизонтали (ширина), либо по вертикали (глубина). Какой бы подход мы ни использовали, увеличение структуры должно повысить общую точность дерева. Формула для соответствующего усиления может быть получена из obj^* . Например, если мы решим разделить лист на два листа, мы можем сформулировать выигрыш следующим образом:

$$Gain = \frac{1}{2} \left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda} \right] - \gamma,$$

где

первое слагаемое – это оценка нового левого листа,

второе слагаемое – оценка нового правого листа,

третье слагаемое – оценка исходного листа,

γ действует как регуляризация для дополнительного листа.

Здесь ветвление произойдет, только если выигрыш больше γ . Реализации повышения градиента могут отличаться тем, как они разбивают интервалы с действительными значениями в узле при создании новых листьев. Это описание завершает этап обучения алгоритма.

После построения окончательной модели ансамбля ее можно использовать для прогнозирования временных рядов в качестве регрессионной модели.

XGBoost. *XGBoost (eXtreme Gradient Boosting)* на данный момент является популярным алгоритмом градиентного бустинга на деревьях. На этапе обучения он дополняет дерево на k -й итерации, разбивая каждый лист и тем самым добавляя еще один уровень дерева (рост в глубину). Значение разделения вычисляется путем предварительной сортировки значений в каждом измерении входных объектов и построения гистограмм. Наилучшее разделение находится путем линейного сканирования по отсортированным значениям признаков. Результирующее дерево более или менее сбалансировано, хотя и асимметрично из-за того, что расщепление происходит только в том случае, если оно приводит к положительному выигрышу. Симметричные деревья были введены в *CatBoost* для оптимизации времени вывода.

LightGBM. *LightGBM (Light Gradient Boosting Machine)* – еще один популярный алгоритм градиентного бустинга. Для многих задач он оказывается быстрее в вычислительном отношении, чем *XGBoost*, обеспечивая при этом ту же точность. Более того, для прогнозирования временных рядов он продемонстрировал превосходные результаты в конкурсе M5 [15]. Его алгоритм увеличения дерева отличается от *XGBoost*. Разделение в *LightGBM* выполняется с использованием односторонней выборки на основе градиента, которая сохраняет экземпляры с большими градиентами и выполняет случайную выборку экземпляров с меньшими градиентами. Это приводит к полистовому росту дерева, где за одну итерацию можно создать несколько уровней для одного начального листа. Это делает дерево менее сбалансированным по сравнению с *XGBoost*. Еще одной функцией *LightGBM* является *Exclusive Feature Bundling*, которая использует тот факт, что функции в разреженном

пространстве функций часто могут быть взаимоисключающими и, таким образом, могут быть объединены в одну функцию.

2.2 Алгоритмы сглаживания

Временные ряды сетевого трафика чрезвычайно зашумлены, поэтому перед применением любого алгоритма прогнозирования требуется дополнительное сглаживание. Ниже мы предоставляем ряд алгоритмов сглаживания и преобразований, которые мы протестировали.

Логарифмирование и преобразование Бокса-Кокса. Это преобразования общего назначения, которые можно применять к любым положительным данным с действительным знаком. Несмотря на то, что они не являются специфическими для данных временных рядов, мы обнаружили, что их использование в качестве процедур предварительной обработки полезно. Для логарифмического преобразования X_t мы использовали натуральный логарифм: $\tilde{X}_t = \ln X_t$. Обратное логарифмическое преобразование: $X = \exp(\tilde{X}_t)$.

Преобразование Бокса-Кокса X_t параметризуется λ и определяется как

$$\tilde{X}_t = \begin{cases} \frac{X_t^\lambda - 1}{\lambda} & \text{if } \lambda \neq 0, \\ \ln X_t & \text{if } \lambda = 0. \end{cases}$$

Тогда обратное преобразование Бокса-Кокса:

$$X_t = \begin{cases} \left(\lambda \tilde{X}_t + 1 \right)^{1/\lambda} & \text{if } \lambda \neq 0, \\ \exp(\tilde{X}_t) & \text{if } \lambda = 0. \end{cases}$$

Заметим, что логарифмическое преобразование является частным случаем преобразования Бокса-Кокса при $\lambda = 0$.

Скользящее среднее. Простое скользящее среднее подразумевает, что наблюдение временного ряда X_t заменяется средним из k предыдущих наблюдений:

$$\tilde{X}_t = \frac{1}{k} \sum_{i=t-k+1}^t X_i.$$

Применение простого скользящего среднего сдвигает временной ряд на $k/2$ запаздывания, что означает, что преобразованный временной ряд должен быть сдвинут назад на такое же количество запаздываний, чтобы устранить эту искусственную задержку.

Экспоненциальное сглаживание. Этот алгоритм можно рассматривать и как алгоритм прогнозирования, и как алгоритм фильтрации. Например, экспоненциальное сглаживание использовалось как фильтрующая часть решения победителя в конкурсе M4 [14]. В качестве алгоритма сглаживания оно записывается в следующем виде:

$$\begin{aligned}\tilde{X}_0 &= X_0, \\ \tilde{X}_t &= \alpha X_t + (1 - \alpha)\tilde{X}_{t-1}, \quad t > 0,\end{aligned}$$

где $\alpha \in (0; 1)$ — коэффициент сглаживания, который следует рассматривать как гипер-параметр.

При фиксированном α обратное преобразование имеет вид:

$$\begin{aligned}X_0 &= \tilde{X}_0, \\ X_t &= \frac{1}{\alpha}\tilde{X}_t - \frac{1 - \alpha}{\alpha}\tilde{X}_{t-1}, \quad t > 0.\end{aligned}$$

С точки зрения обработки сигналов и экспоненциальное сглаживание, и скользящее среднее являются фильтрами нижних частот. Экспоненциальное сглаживание, в свою очередь, представляет собой взвешенное скользящее среднее с экспоненциально затухающими весами.

Анализ сингулярного спектра (SSA). Среди других приложений SSA можно использовать для разложения временного ряда на сумму компонентов, обычно имеющих интерпретируемое значение (например, тренд или периодические колебания). Данное преобразование основано на теореме вложения с задержкой (теорема Такена), говорящей о том, что ключевые свойства динамической системы могут быть восстановлены в пространстве вложения с задержкой. Последнее получается преобразованием одномерного временно-

го ряда в многомерный путем взятия его первых L лагов. Чтобы разложить временной ряд, SSA использует разложение по сингулярным числам (SVD) в таком пространстве вложения задержки.

Пусть F_t обозначает временной ряд с $t = 0, \dots, N - 1$, а L обозначает количество лагов, используемых для построения пространства вложения задержек. Для удобства мы также определяем $K = N - L + 1$. Затем каждое значение F_t можно преобразовать в вектор $[F_t, F_{t+1}, \dots, F_{t+L-1}]^T$. Это дает матрицу траекторий \mathbf{X} :

$$\mathbf{X} = \begin{bmatrix} F_0 & F_1 & \dots & F_{N-L} \\ F_1 & F_2 & \dots & F_{N-L+1} \\ & & \dots & \\ F_{L-1} & F_L & \dots & F_{N-1} \end{bmatrix}$$

которая имеет ганкелевую (антидиагональную) структуру, т. е. имеет равные значения на антидиагоналях.

Затем матрица траектории раскладывается с помощью SVD :

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T,$$

где

\mathbf{U} является $L \times L$ унитарной матрицей, столбцы которой представляют собой левые сингулярные векторы \mathbf{X} ;

$\mathbf{\Sigma}$ представляет собой прямоугольную диагональную матрицу $L \times K$, диагональные элементы которой представляют собой L сингулярные значения \mathbf{X} , упорядоченные от наибольшего к наименьшему;

\mathbf{V} является $K \times K$ унитарной матрицей, столбцы которой являются правыми сингулярными векторами матрицы \mathbf{X} .

Это также может быть записано как

$$\mathbf{X} = \sum_{i=0}^{d-1} \sigma_i \mathbf{u}_i \mathbf{v}_i^T = \sum_{i=0}^{d-1} \mathbf{X}_i,$$

где

$\sigma_i - i^{th}$ сингулярное значение;

$\mathbf{u}^{(i)}$ и $\mathbf{v}^{(i)}$ – левые и правые сингулярные векторы, т. е. i^{th} -столбцы \mathbf{U} и \mathbf{V} ;

$d \leq L$ – ранг матрицы траекторий.

Здесь \mathbf{X}_i называется i^{th} элементарной матрицей \mathbf{X} . Элементарные матрицы \mathbf{X}_i могут иметь интерпретируемый смысл. Например, мы могли бы сгруппировать их как элементарные матрицы «тренда» $\{\mathbf{X}_j\}_{j \in T}$, элементарные матрицы «сезонности» $\{\mathbf{X}_j\}_{j \in S}$ и другие как шум, где T и S – соответствующие наборы индексов. Затем мы могли бы выполнить сглаживание, просто взяв матрицы тренда и сезонности. Однако для этого требуется, чтобы каждая из этих матриц имела структуру Ганкеля, подобную исходной матрице траектории, что практически невозможно гарантировать для реальных временных рядов. Таким образом, каждая элементарная матрица должна быть преобразована в матрицу Ганкеля, что эквивалентно извлечению временного ряда из элементарной матрицы.

Для этой цели воспользуемся диагональным усреднением. Оно определяет значения восстановленного временного ряда \tilde{F}^j как средние значения соответствующих антидиагоналей элементарной матрицы \mathbf{X}^j . Формально это выражается оператором ганкелизации H , который действует на $L \times K$ -матрицу \mathbf{X}^j и дает ганкелеву матрицу $\tilde{\mathbf{X}}^j$, элементы которой $\tilde{x}_{m,n}$ для $s = m + n$ задаются формулой:

$$\tilde{x}_{m,n} = \begin{cases} \frac{1}{s+1} \sum_{l=0}^s x_{l,s-l}, & 0 \leq s \leq L-1 \\ \frac{1}{L-1} \sum_{l=0}^{L-1} x_{l,s-l}, & L \leq s \leq K-1 \\ \frac{1}{K+L-s-1} \sum_{l=s-K+1}^L x_{l,s-l}, & K \leq s \leq K+L-2 \end{cases}$$

Наконец, мы можем сгруппировать элементарные матрицы и соответствующие реконструированные временные ряды на основе их близости. Например, полученные группы можно связать с тенденцией и сезонностью. Определим взвешенное скалярное произведение для двух восстановленных

временных рядов \tilde{F}_i и \tilde{F}_j как $\langle \tilde{F}_i, \tilde{F}_j \rangle_w = \sum_{k=0}^{N-1} w_k \tilde{f}_{i,k} \tilde{f}_{j,k}$, где $\tilde{f}_{i,k}$ и $\tilde{f}_{j,k}$ — k^{th} значения \tilde{F}^i и \tilde{F}^j . Веса w_k определяются следующим образом:

$$w_k = \begin{cases} k + 1, & 0 \leq k \leq L - 1 \\ L, & L \leq k \leq K - 1 \\ N - k, & K \leq k \leq N - 1 \end{cases}$$

Вес w_k просто отражает количество раз, когда $\tilde{f}_{i,k}$ появляется в соответствующей ганкелизованной матрице \tilde{X}^i , из которых получен временной ряд \tilde{F}^i . Будем говорить, что компоненты временных рядов \tilde{F}^i и \tilde{F}^j отделены, если они w -ортогональны, т. е. $\langle \tilde{F}^i, \tilde{F}^j \rangle_w = 0$. Поскольку истинная w -ортогональность не встречается в реальных временных рядах, удобно ввести коэффициент корреляции $W_{i,j}$, который можно рассматривать как меру подобия:

$$W_{i,j} = \frac{\langle \tilde{F}_i, \tilde{F}_j \rangle_w}{\|\tilde{F}_i\|_w \|\tilde{F}_j\|_w}, \quad \text{где} \quad \|\tilde{F}_k\|_w = \sqrt{\langle \tilde{F}_k, \tilde{F}_k \rangle_w}.$$

Интерпретация $W_{i,j} \in [0; 1]$ проста: чем ближе вес к 1, тем больше похожи \tilde{F}^i и \tilde{F}^j . Как правило, мы можем сказать, что $W_{i,j} \gtrsim 0.3$ означает, что \tilde{F}^i и \tilde{F}^j должны быть сгруппированы вместе.

2.3 Алгоритм валидации

Временные ряды сетевого трафика могут включать в себя сегменты с очень большими или малыми значениями, динамика которых не следует из предшествующей истории. Включение этих сегментов в обучающий набор данных без адаптации модели снизит точность, что показано на рисунке 5. Поэтому мы должны адаптировать модель прогнозирования, чтобы она могла переключаться между различными динамическими режимами.

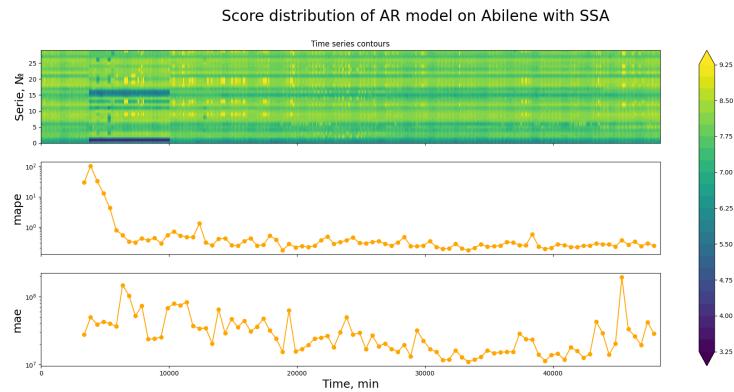


Рис. 5: Распределение метрик качества относительно резких изменений динамики временных рядов.

Производительность модели прогнозирования сильно зависит от валидации, поскольку она определяет, как набор данных будет разделен для обучения и проверки и какие значения гипер-параметров будут выбраны. Существует ряд установленных способов валидации для классификации, кластеризации и регрессии, но валидация прогнозирования временных рядов отличается: «выборки» не являются статистически независимыми (iid) и в этом случае явно упорядочены, т. е. перетасовка никогда не будет вариантом для них. Учитывая отсутствие надлежащего объяснения процедур проверки в исследовательских работах, мы решили дать подробное описание алгоритма проверки, который мы разработали для нашего исследования.

Наш алгоритм проверки основан на скользящем окне. Предположим, что X_t — это весь многомерный временной ряд, который у нас есть в качестве набора данных для прогнозирования. Затем алгоритм валидации гласит:

1. Выберите размер окна обучения T_{train} , размер тестового окна T_{test} (он должен быть равен ожидаемому временному горизонту прогноза) и временной интервал T_{gap} (это значение помогает контролировать вычислительную сложность валидации).
2. Для каждого временного ряда X_t сгенерируйте последовательность временных рядов $X_{t,k}$, такую, что $X_{0,k} = X_{kT_{gap}}$, а длина $X_{t,k}$ равна $T_{train} + T_{test}$.
3. Для каждого $X_{t,k}$ обучите рассматриваемую модель на временных рядах $X_{0:T_{train},k}$ и проверьте на $X_{T_{train}:T_{test},k}$.
4. Зафиксируйте средний результат проверки среди всех временных рядов $X_{t,k}$.

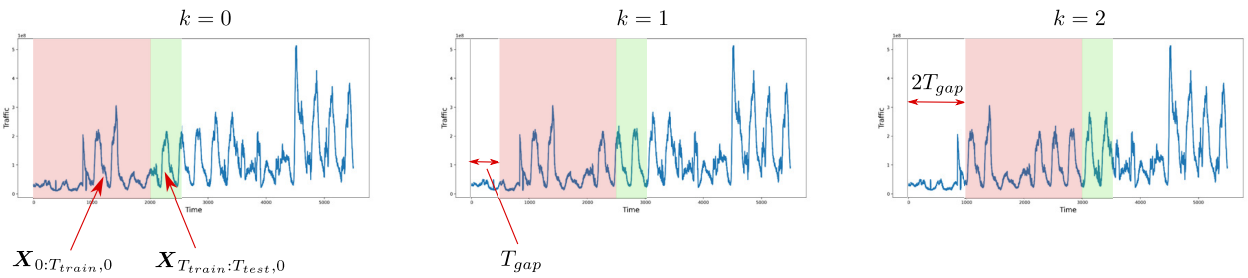


Рис. 6: Иллюстрация процедуры валидации для $k = 0, 1, 2$.

Для реализации этой процедуры мы используем библиотеку `sklearn` и ее класс `TimeSeriesSplit` из `sklearn.model_selection`, который просто разбивает временной ряд на части `n_splits`. Таким образом, мы указываем `n_split` вместо T_{gap} , который можно легко вычислить из `n_split`.

Далее приведен пример разделения нашим алгоритмом для временного ряда $[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]$ и $T_{train} = 3, T_{test} = 2$:

1 итерация:

обучающая выборка: $[0, 1]$

тестовая выборка: $[2, 3]$

2 итерация:

обучающая выборка: [1, 2, 3]

тестовая выборка: [4, 5]

3 итерация:

обучающая выборка: [3, 4, 5]

тестовая выборка: [6, 7]

Финальная итерация:

обучающая выборка: [5, 6, 7]

тестовая выборка: [8, 9]

Обычно мы пропускаем первое разбиение, потому что оно может сократить окно обучения, которое становится слишком маленьким для обучения некоторых моделей.

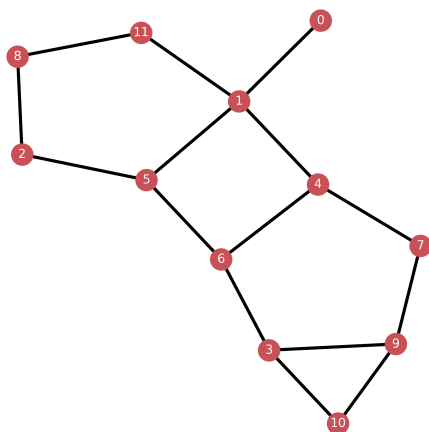
Глава 3. Анализ данных

Для обучения и тестирования наших моделей нам нужны наборы данных, которые содержат достаточное количество наблюдений и имеют графовую структуру. В данной работе мы изучили три общедоступных датасета, связанных с трафиком: *Abilene*, *Totem* и *PeMSD7*. Они содержат наборы временных рядов, связанных с узлами или парами узлов соответствующих топологий. Прежде всего, важно исследовать эти временные ряды с точки зрения их пространственных, временных и статистических характеристик. Отметим, что вышеупомянутые наборы данных содержат временные ряды, определенные на сетях, что нетипично для стандартного анализа данных. Для каждого из них мы подготовили краткое описание и провели корреляционный анализ, чтобы изучить зависимости между узлами сети.

3.1 Abilene

Abilene [25] – это магистральная сеть в Северной Америке, состоящая из 12 маршрутизаторов, для которой в течение 24 недель за 2004 год собирались объемы трафика (*байт/с*) потоков отправления-назначения, агрегированные пятиминутным интервалом.

Соответствующая топология и примеры временных рядов входящего трафика показаны на рисунках 7а, 7б. Нетрудно заметить, что эти временные ряды сильно зашумлены, имеют аномальные выбросы и временные интервалы с резкой сменой динамики. Тем не менее, они демонстрируют сезонность и краткосрочные лаговые корреляции, что дает возможность сделать разумный краткосрочный прогноз.



(a) Топология Abilene



(b) Примеры временных рядов из набора данных Abilene

Рис. 7: Обзор набора данных Abilene

На рисунке 8а показана корреляционная матрица для входного трафика, связанного с каждым конкретным узлом. Чтобы сделать рисунок разреженным, мы обнулили те коэффициенты корреляции, абсолютное значение которых меньше 0.3. Картина корреляции лишь частично соответствует структуре сети: из 15 ребер только 7 сопровождаются значительными корреляциями, которые можно увидеть на рисунке 8б. При этом большинство корреляционных связей нельзя отнести к сильным: только узлы 5 и 6 имеют сильную корреляцию ($\rho \approx 0.81$). С другой стороны, узлы 3 и 5 также имеют высокую корреляцию, но они не являются соседями. Для наглядности мы построили график для *Abilene*, используя корреляционную матрицу (Рис. 8б), где серые ребра обозначают действительные связи, а синие – корреляционные.

Чтобы глубже изучить пространственные зависимости трафика (или заявить об их отсутствии), сначала рассмотрим распределение коэффициентов корреляции между узлами и их соседями. На рисунке 9 показаны эти распределения для каждого узла. Они предполагают отсутствие сильной пространственной корреляции: половина узлов не имеет значимых корреляций со своими соседями. Чтобы подкрепить этот вывод, мы дополнительно вычисляем зависимость коэффициентов корреляции от расстояния кратчайшего пути между узлами, как показано на рисунке 10. Несмотря на то, что количество узлов в топологии *Abilene* слишком мало, чтобы сделать однозначный вывод, ясно, что сильной зависимости нет.

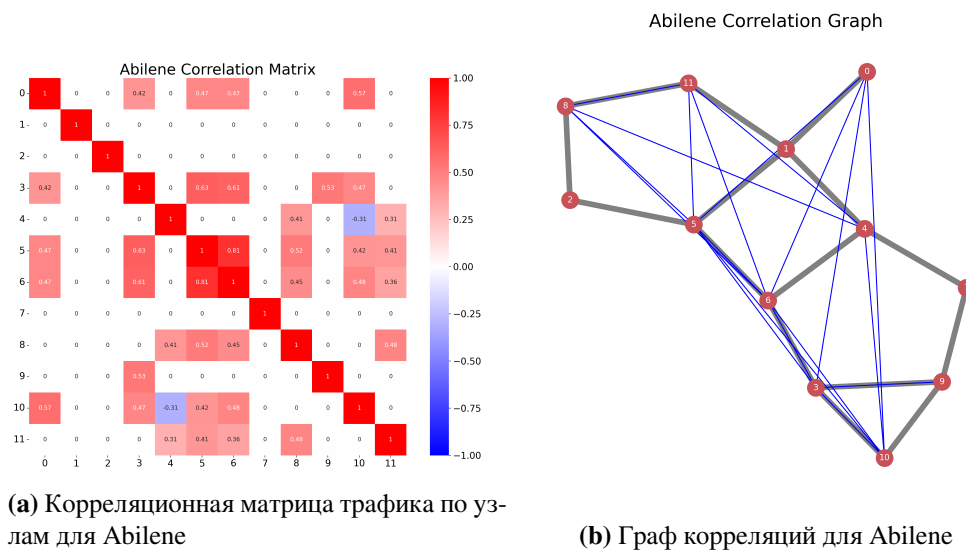


Рис. 8: Корреляционные зависимости Abilene

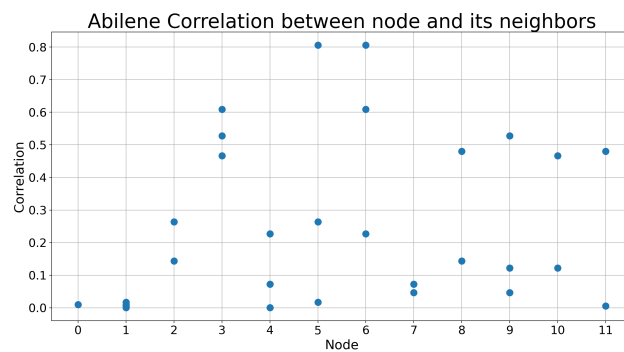


Рис. 9: Корреляция между узлами и их соседями для Abilene

Как правило, системы с пространственно-временными зависимостями демонстрируют убывающую корреляцию в зависимости от расстояния между двумя точками («расстояние» следует понимать в общих чертах, поскольку сети определяют неевклидово пространство). Каноническим примером является двухточечная корреляция в турбулентных течениях (см. раздел 6.3 в [18]). Отсутствие затухающей корреляции в нашем случае ставит под сомнение наличие пространственной зависимости во временном ряду трафика.

В целом объяснений этому наблюдению может быть три:

1. Никакой пространственной зависимости нет. Если это объяснение верно, прогнозирование временных рядов должно быть одномерным, т. е. каждый временной ряд можно прогнозировать отдельно.

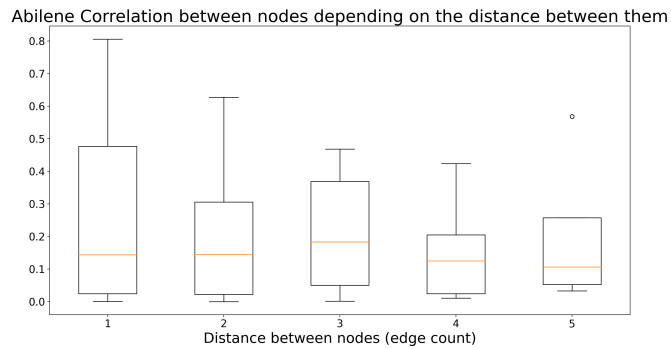


Рис. 10: Корреляция узлов в зависимости от расстояния между ними для Abilene

2. Пространственная зависимость существует: она нелинейна и не может быть отражена коэффициентом корреляции Пирсона. Если это объяснение верно, многомерное прогнозирование временных рядов должно быть лучше одномерного, которое мы должны наблюдать во время сравнительного анализа.
3. Пространственная зависимость существует, но в меньших масштабах времени. В этом случае мы могли бы утверждать, что сетевая система уравнивается в масштабах времени намного короче 5 минут. Аналогичная ситуация наблюдается и на финансовых рынках, где она носит название *пространственный арбитраж*. Если это объяснение верно, стоит изучить механизм, посредством которого происходит арбитражный процесс, поскольку он прокладывает путь к теоретико-игровому подходу к прогнозированию и контролю.

После изучения корреляций мы можем перейти к исследовательскому анализу данных отдельных временных рядов трафика. На рисунках 11, 12, 13 показаны гистограммы, автокорреляционные функции (*ACF*) и частичные автокорреляционные функции (*PACF*) для выбранных репрезентативных логарифмически преобразованных временных рядов. Логарифмирование необходимо использовать как для анализа, так и для прогнозирования, поскольку известно, что интернет-трафик подчиняется логарифмически нормальному распределению [1]. Оно приводит к нормальному распределению, которое, по общему признанию, благоприятно для широкого круга численных методов. Гистограммы на рисунках 11а, 12а, 13а действительно подтверждают, что

результатирующее распределение является нормальным по модулю временных средних сдвигов, ведущих к бимодальности.

Еще одно важное сообщение можно увидеть из автокорреляционных функций. Для некоторых временных рядов (Рисунки 12b, 13b) ACF характеризуются сильной периодичностью, что предполагает сезонность в соответствующих временных рядах с периодом 288 (ровно сутки). Эта закономерность, однако, проявляется не во всех временных рядах: например, она может быть затенена трендом, даже если мы не наблюдаем сильных тенденций в данных.

Наконец, частичные автокорреляционные функции (Рисунки 11с, 12с, 13с) ясно указывают на то, что истинные корреляции существуют только для одного лага, т. е. они соответствуют процессу $AR(1)$, тогда как другие, вероятно, объясняются сезонностью и трендом.

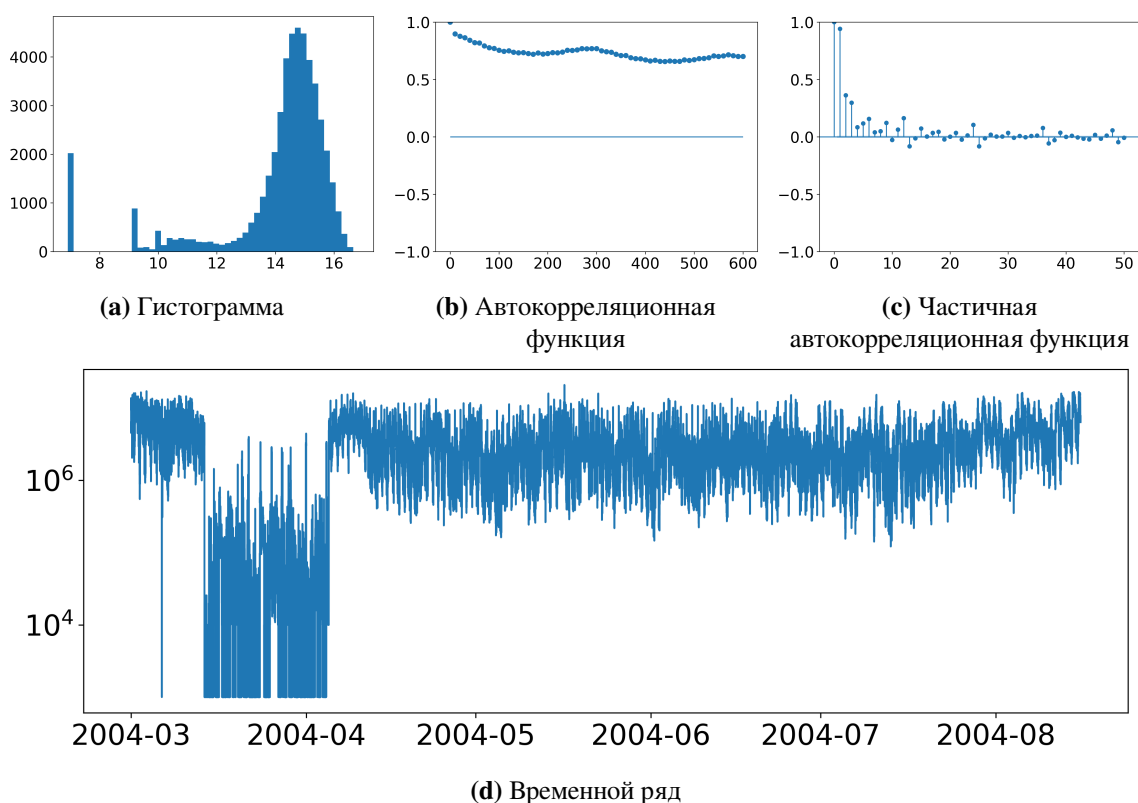


Рис. 11: Исследовательский анализ данных для 0 узла Abilene

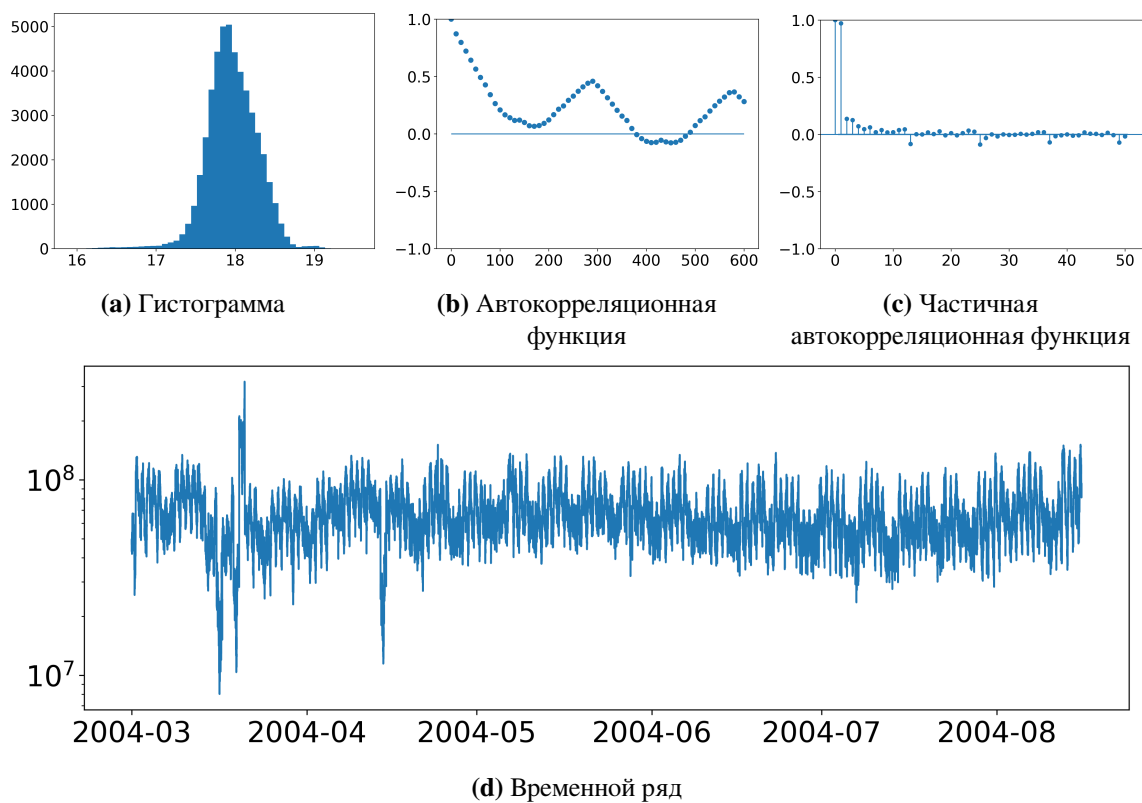


Рис. 12: Исследовательский анализ данных для 3 узла Abilene

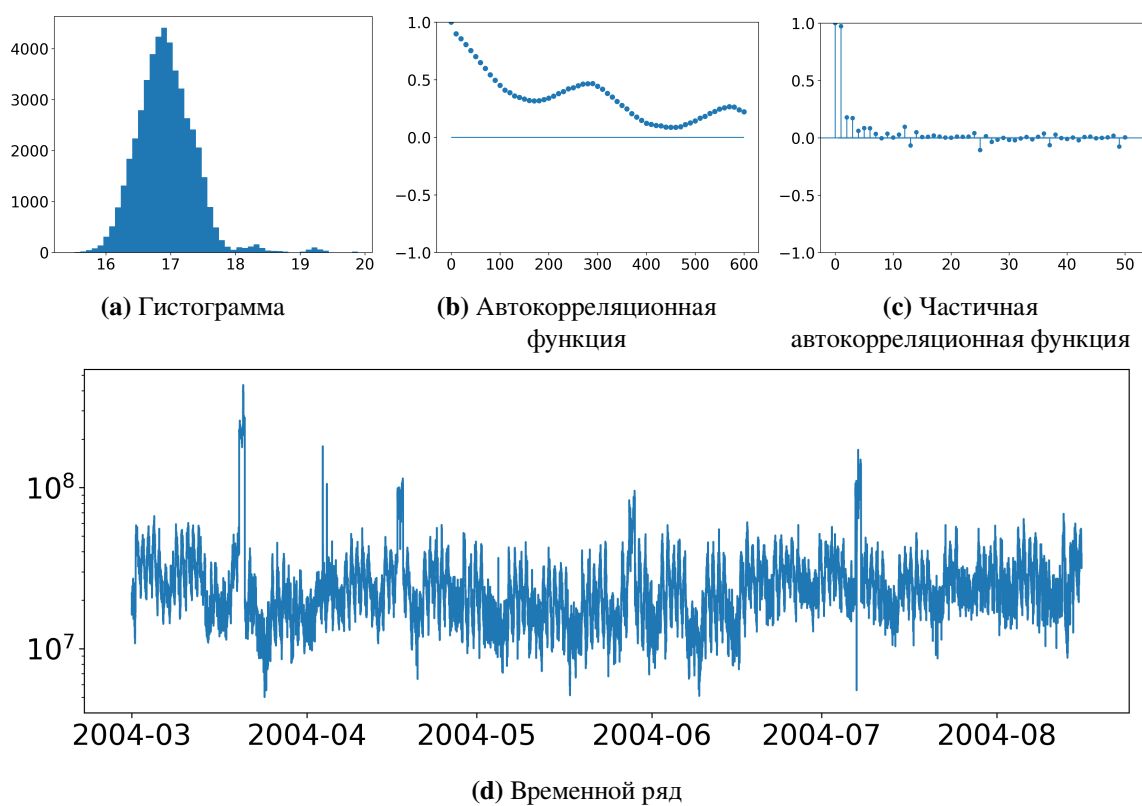
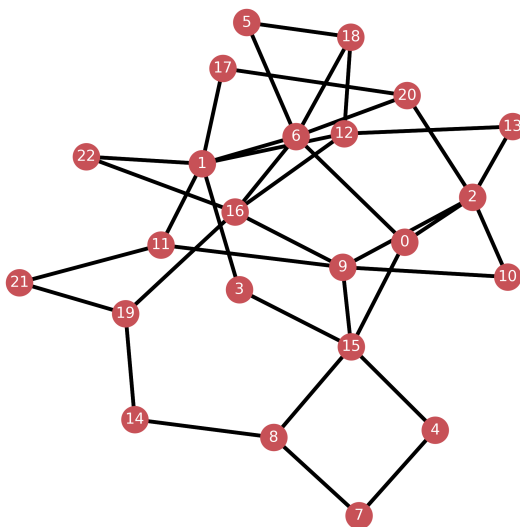


Рис. 13: Исследовательский анализ данных для 9 узла Abilene

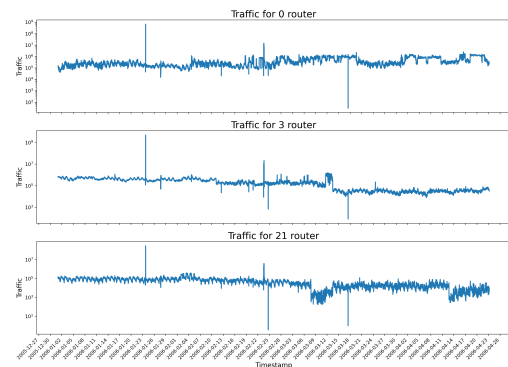
3.2 Totem

Totem GEANT [8] – общеевропейская исследовательская сеть, управляемая *DANTE*. Он обрабатывает весь трафик от европейских национальных исследовательских и образовательных сетей (*NREN*), соединяющих университеты и исследовательские институты. *GEANT* имеет *PoP* (точку присутствия) в каждой европейской стране. Все маршрутизаторы *GEANT* являются пограничными. *Totem* состоит из 23 узлов, соединенных между собой 74 каналами. Набор данных охватывает около 4 месяцев (январь-апрель 2005 г.) с шагом 15 минут и соответствует объему входного трафика, измеряемому в *Кбайт/с*.

Топология и отдельные примеры временных рядов трафика показаны на рисунках 14а, 14б. Этот набор данных похож на *Abilene*, но имеет больше маршрутизаторов и более неравномерную динамику, включая экстремальные выбросы и изменения амплитуды временных рядов.



(а) Топология Totem

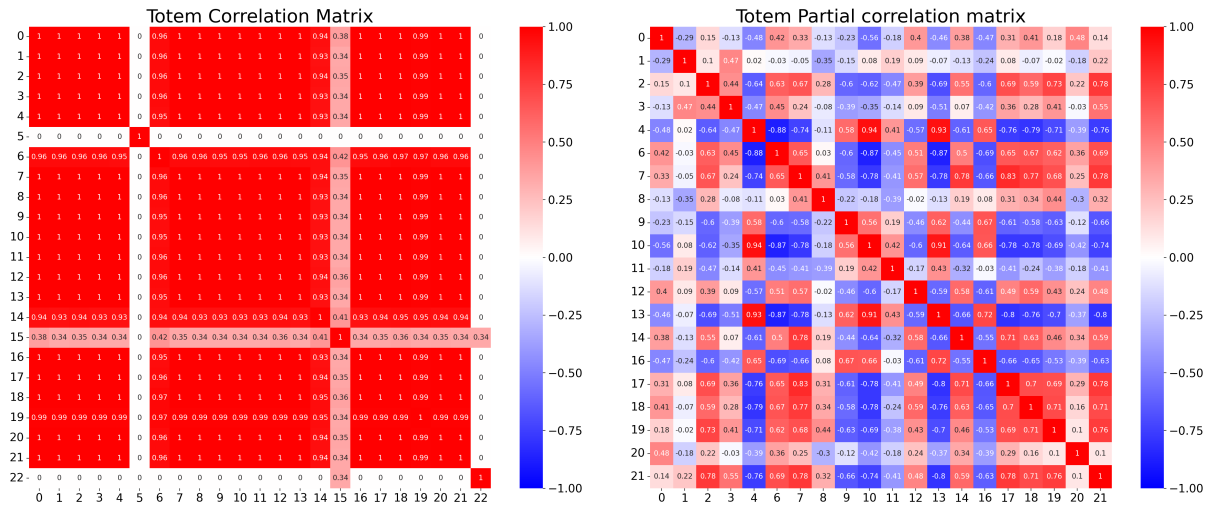


(б) Примеры временных рядов из набора данных Totem

Рис. 14: Обзор набора данных Totem

В отличие от набора данных *Abilene*, *Totem* имеет аномально высокие значения корреляции, что видно из рисунка 15а. Таким образом, временные ряды выглядят как дубликаты: большинство из них имеют общие выбросы и общую закономерность. Мы предполагаем, что эта динамика управляется

общей динамикой трафика. Чтобы получить корреляции узлов по модулю общего трафика, мы дополнительно вычисляем частичные корреляции, учитывая средний трафик всех узлов (за исключением узла 5 и узла 22, которые не коррелируют с другими) и отображаем их на рисунке 15b. Несмотря на более разреженную матрицу, ее значения по-прежнему предполагают сильную линейную зависимость между большинством узлов.



(a) Корреляционная матрица для Totem

(b) Матрица частичной корреляции для Totem

Рис. 15: Матрицы корреляций Totem для входящего трафика по узлам

Подобно набору данных *Abilene*, мы провели анализ пространственной зависимости, вычислив корреляцию между узлами и их соседями (рисунок 16) и ее зависимость от расстояния между узлами (рисунок 17). Его результаты, однако, зашумлены высокими значениями коэффициента корреляции, и поэтому их трудно интерпретировать.

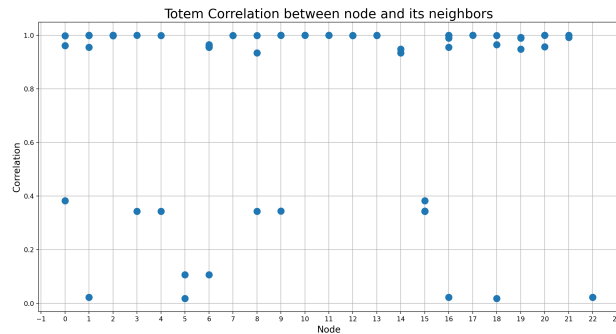


Рис. 16: Корреляция между узлами и их соседями для Totem

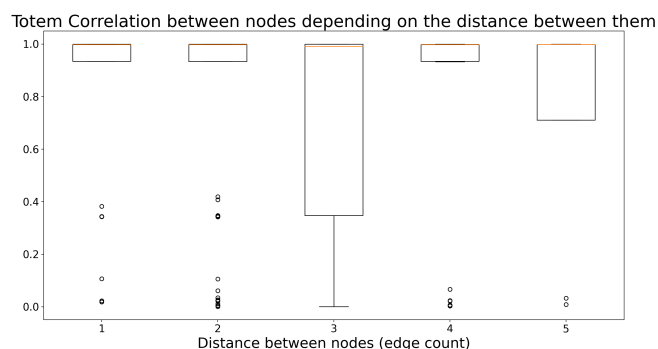


Рис. 17: Корреляция узлов в зависимости от расстояния между ними для Totem

Теперь проанализируем трафик отдельных временных рядов, показанных на рисунках 18, 19, 20 (в данном случае анализируются логарифмически преобразованные временные ряды). Набор данных *Totem* более зашумлен и содержит серьезные средние сдвиги (Рисунок 18d) и выбросы. Таким образом, трудно ожидать, что предположение о нормальности будет выполняться для всех временных рядов: гистограммы показывают бимодальность (Рисунок 19a) и асимметрию (Рисунки 18a, 20a) для некоторых временных рядов. Обратите внимание, что узлы 5, 15 и 22 имеют исключительно низкие коэффициенты корреляции на рисунке 15a.

Автокорреляционные функции также отражают разнообразие характеристик временных рядов. Для узла 5 (Рисунок 18b) *ACF* демонстрирует сильную сезонность с однодневным периодом. Узлы 15 и 22 имеют тенденцию, которая приводит к медленному затуханию *ACF* (Рисунки 19b, 20b).

Частичные автокорреляционные функции (Рисунки 18с, 19с, 20с) показывают, что истинные корреляции существуют только для одного лага, также как и в случае с набором *Abilene*.

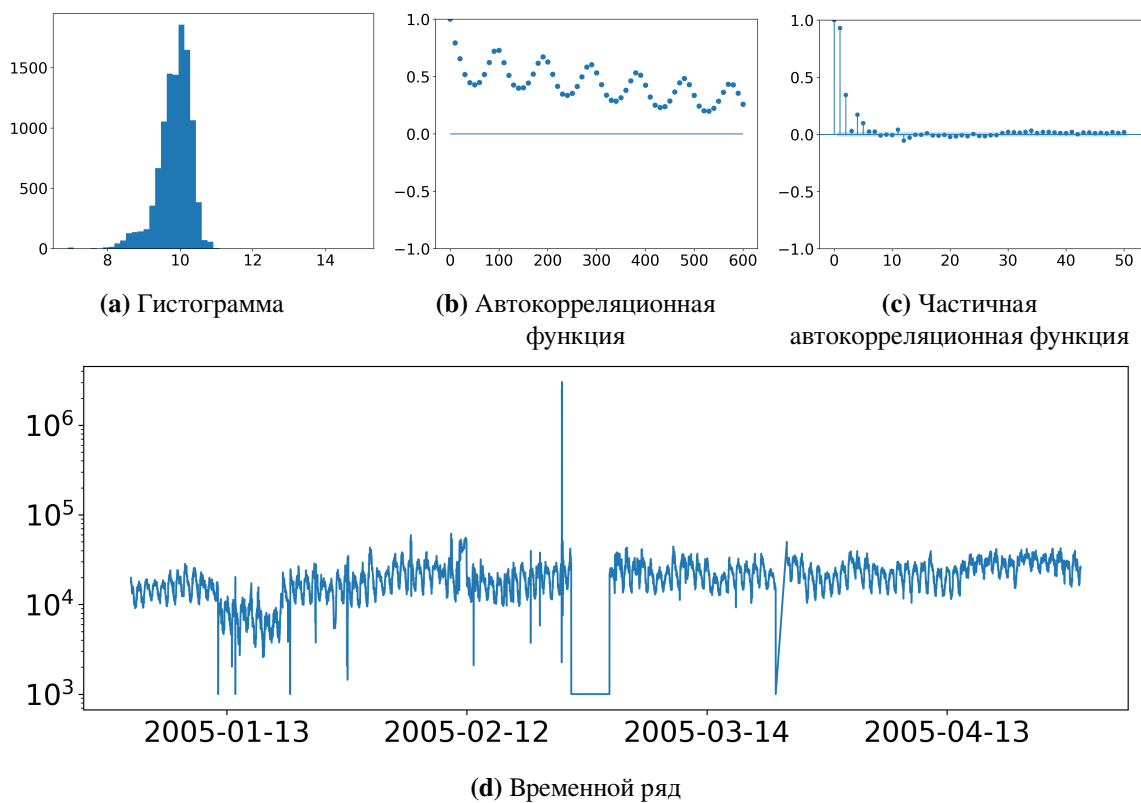


Рис. 18: Исследовательский анализ данных для 5 узла Totem

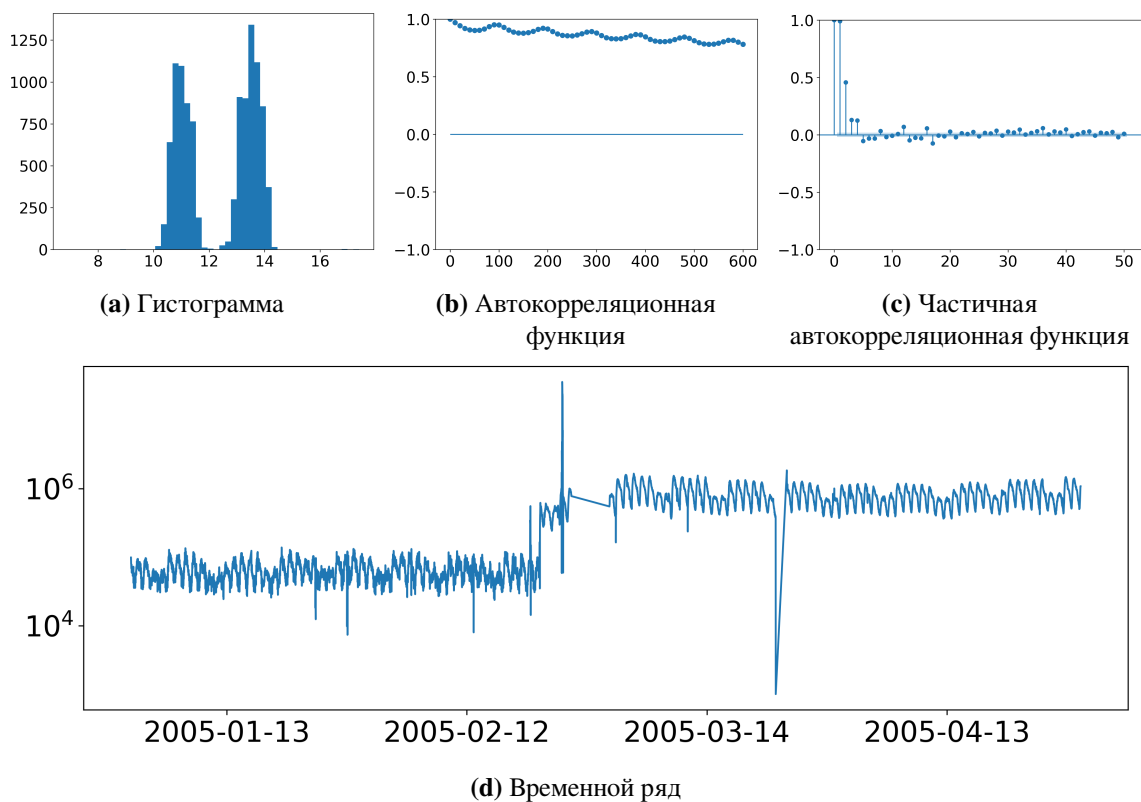


Рис. 19: Исследовательский анализ данных для 15 узла Totem

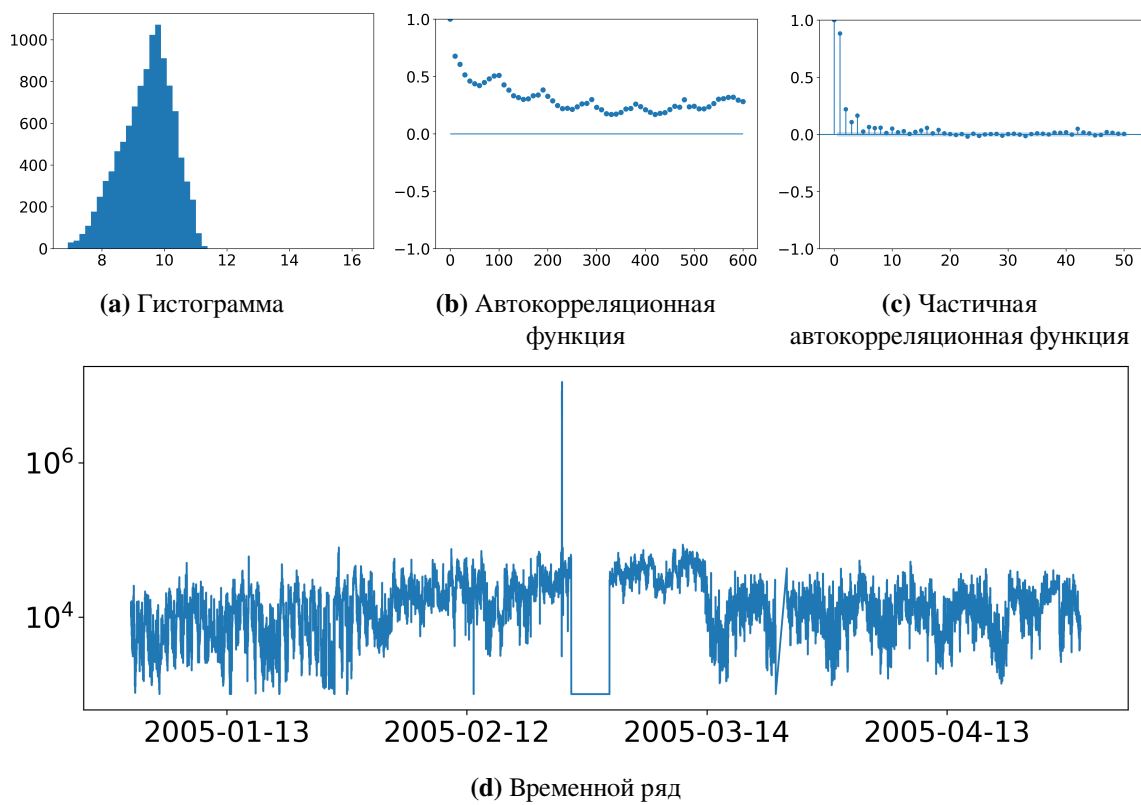


Рис. 20: Исследовательский анализ данных для 22 узла Totem

3.3 PeMSD7

Исходные данные для этого набора были собраны из системы измерения загруженности трафика *Caltrans (PEMS)* в режиме реального времени с помощью более чем 39000 сенсорных станций системы автомагистралей штата Калифорния. Набор данных также объединяется в 5-минутные интервалы из 30-секундных выборок. Чтобы уменьшить объем датасета, для создания набора *PeMSD7* [22] были случайным образом выбраны только 228 станций в седьмом округе штата Калифорния. Набор данных охватывает рабочие дни мая и июня 2012 года и соответствует скорости, измеренной в милях в час.

Этот набор данных не предоставляет топологию напрямую. Вместо этого он содержит матрицу расстояний $D = [d_{i,j}]$ для дорог, соединяющих «узлы», которую мы можем использовать для вычисления матрицы смежности $A = [A_{i,j}]$, определяющей топологию. А именно, мы используем следующее правило, отфильтровывая крошечные участки дорог:

$$A_{i,j} = \begin{cases} 1, & \text{если } i \neq j \text{ и } \exp(\frac{-d_{i,j}^2}{\sigma^2}) \geq \varepsilon, \\ 0, & \text{иначе,} \end{cases}$$

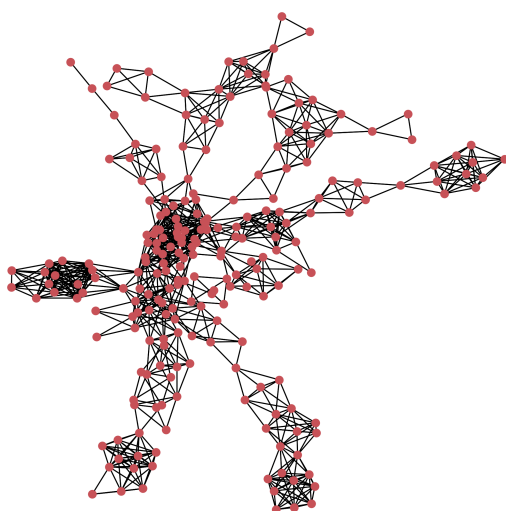
где

σ – параметр, определяющий ширину ядра Гаусса, примененного к матрице квадратных расстояний,

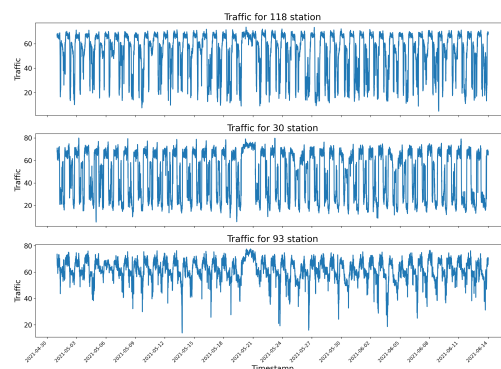
ε – порог, указывающий, есть ли ребро (дорога) между двумя узлами.

Соответствующая топология и избранные примеры временных рядов показаны на рисунках 21a, 21b. Временные ряды в этом наборе данных гораздо более детерминированы (т. е. менее зашумлены) по сравнению с двумя предыдущими наборами сетевых данных, что проявляется в ярко выраженной периодичности. У них также гораздо меньше выбросов.

Подобно набору данных *Totem*, корреляционная матрица для трафика *PeMSD7* имеет значения, близкие к единице за некоторыми исключениями с нулевыми коэффициентами (см. рисунок 22).



(a) Топология PeMSD7



(b) Примеры временных рядов из набора данных PeMSD7

Рис. 21: Обзор набора данных PeMSD7

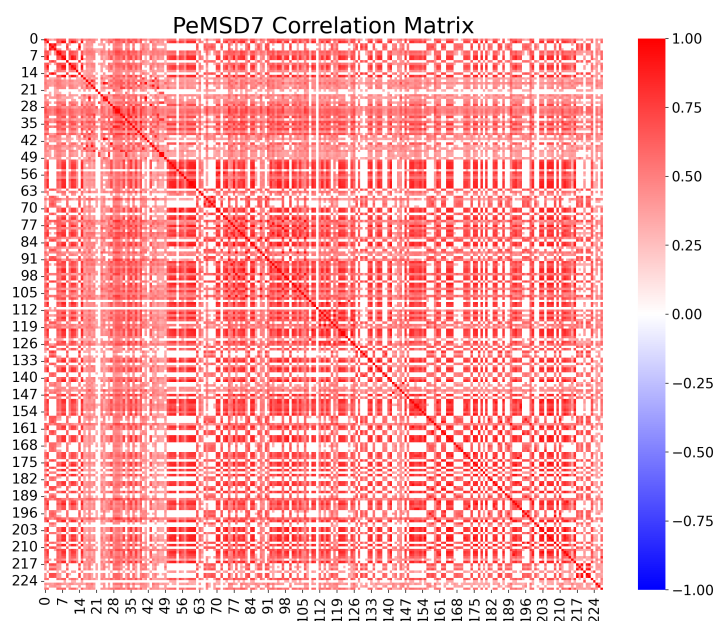


Рис. 22: Корреляционная матрица трафика по узлам для PeMSD7

Благодаря большому количеству узлов этот набор данных особенно подходит для пространственного анализа. Тем не менее, мы по-прежнему не наблюдаем зависимости распределения коэффициента корреляции от расстояния между узлами, что показано на рисунке 23. Медианная корреляция составляет около 0.5 почти для всех расстояний. Мы будем игнорировать большие зависимости на расстоянии 15 и 17, так как это выглядит как ано-

малая. Наибольшая зависимость наблюдается на расстоянии 1, то есть среди ближайших соседей.

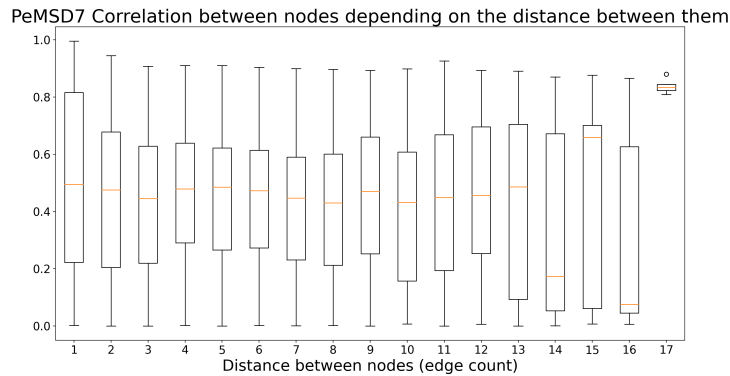


Рис. 23: Корреляция узлов в зависимости от расстояния между ними для PeMSD7

Анализ данных отдельных временных рядов трафика с логарифмическим преобразованием можно найти на рисунках 24, 25, 26. По сравнению с *Abilene* и *Totem* этот набор данных более детерминирован и однороден: все временные ряды демонстрируют сильную сезонность с однодневным периодом и отсутствием каких-либо трендов (Рисунки 24b, 25b, 26b). Для этих временных рядов также характерно бимодальное распределение (Рисунки 24a, 25a): более низкая мода соответствует движению в час пик, а более высокая – ограничению скорости на конкретной дороге. Частичные автокорреляционные функции (Рисунки 24с, 25с, 26с) показывают, что ряды должны быть хорошо аппроксимированы процессом $AR(1)$ в предположении линейности.

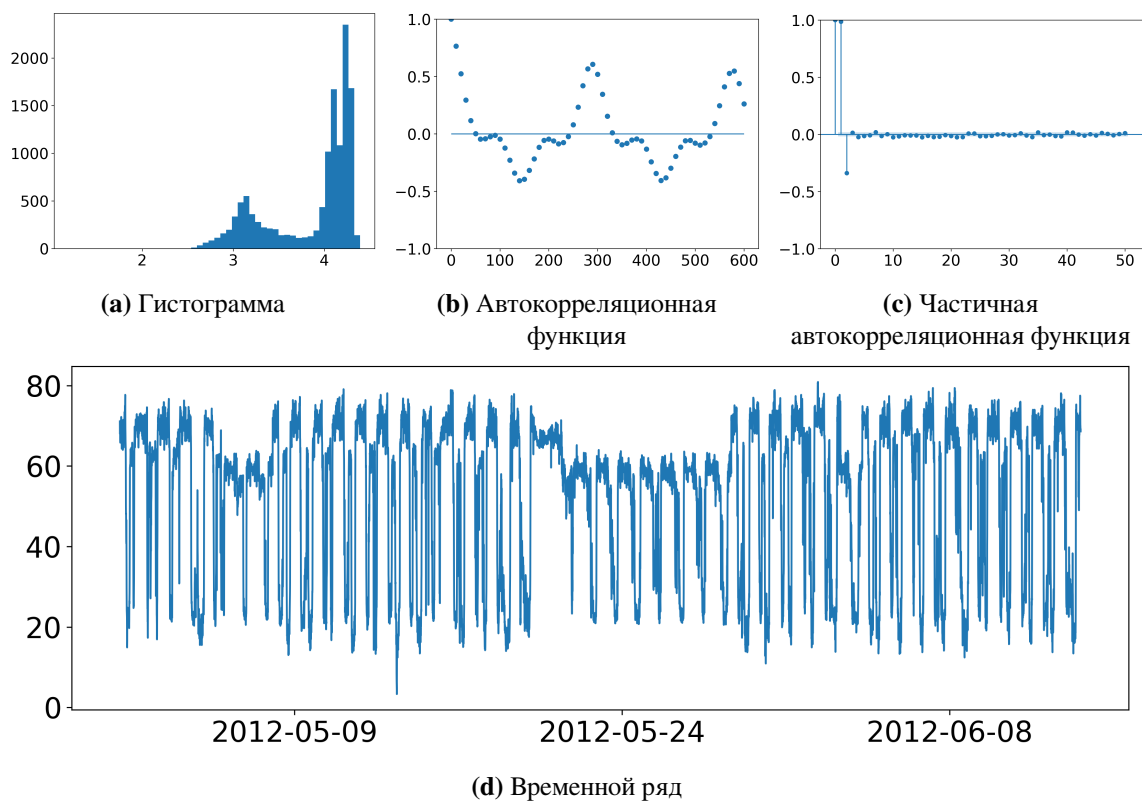


Рис. 24: Исследовательский анализ данных для 0 узла ReMSD7

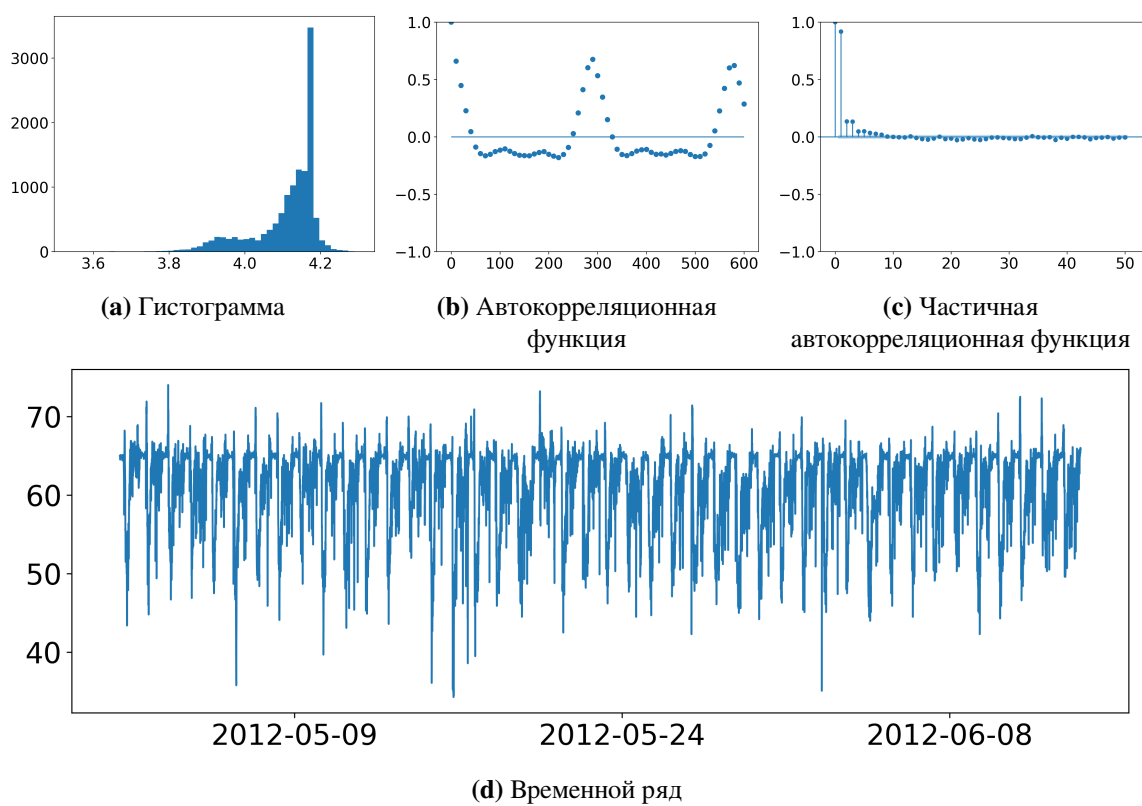


Рис. 25: Исследовательский анализ данных для 62 узла ReMSD7

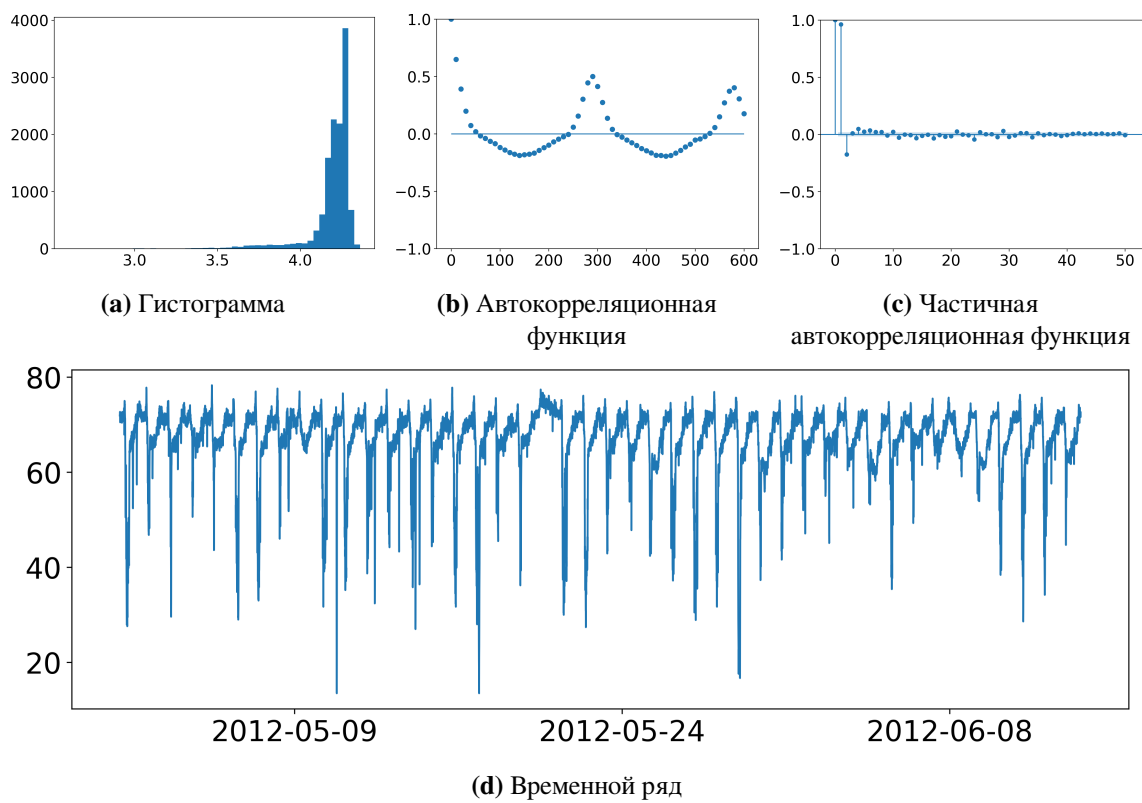


Рис. 26: Исследовательский анализ данных для 157 узла ReMSD7

3.4 Вывод

Мы изучили три открытых набора данных: *Abilene*, *Totem* и *PeMSD7*. Все они содержат набор временных рядов, определенных в сети, топология которой была предоставлена для каждого набора данных. *Abilene* и *Totem* содержат временные ряды интернет-трафика, собранные из магистральных сетей, в то время как *PeMSD7* характеризуется дорожным трафиком. Основываясь на нашем исследовательском анализе этих наборов данных, можно сформулировать следующие выводы:

- В отличие от дорожного движения, обычно изучаемого в исследованиях, посвященных прогнозированию временных рядов, интернет-трафик сильно зашумлен, имеет аномальные выбросы и временные интервалы с резкой сменой динамики.
- В то же время интернет-трафик – это не белый шум и не случайное блуждание. Он явно имеет автокорреляции на ближайших лагах и определенную сезонность (как правило, суточную сезонность), что делает возможным более-менее точное краткосрочное прогнозирование.
- При работе с данными о трафике необходимо использовать логарифмическое преобразование.
- Корреляции между различными узлами в данных о трафике не обязательно отражают топологию соединения. Это должно ограничивать эффективность алгоритмов многомерного прогнозирования, основанных на пространственных зависимостях.
- Трафик может быть сильно коррелирован для всех узлов из-за общего шаблона, который они все разделяют (например, тренды или колебания).

Таким образом, можно сделать вывод, что для многомерного прогнозирования следует использовать данные от ближайших соседей с расстоянием в одно ребро, поскольку именно корреляции с ближайшими соседями имели наибольшие значения для рассмотренных датасетов.

Глава 4. Вычислительные эксперименты

Мы протестировали ряд комбинаций алгоритмов сглаживания и прогнозирования на доступных наборах данных. Кроме того, мы стремились изучить влияние пространственной информации на точность моделей. По умолчанию алгоритмы прогнозирования, которые мы рассматриваем, используются для одномерного предсказания временных рядов, поэтому для данной цели мы вводим их многомерные вариации по ближайшим соседям. Для всех алгоритмов и наборов данных мы прогнозируем на $T_{test} = 500$ шагов вперед.

4.1 Подготовка среды

Все алгоритмы, описанные в главе 2, были реализованы на языке программирования Python с использованием следующих библиотек:

- statsmodels для статистических моделей, таких как *AR*, *VAR*, *Holt-Winters*;
- lightgbm для модели *LightGBM*;
- xgboost для модели *XGBoost*;
- prophet для модели *Prophet*;
- tensorflow для нейросетевых моделей;
- sklearn для создания собственных классов с перечисленными моделями прогнозирования, их обучения, тестирования и настройки параметров;
- pandas, numpy, pickle для обработки данных и их анализа;
- matplotlib для визуализации результатов;
- hydra для удобной работы с конфигурационными файлами;
- и другие библиотеки.

4.2 Тестирование и выбор основных методов

Результаты тестирования моделей для различных алгоритмов прогнозирования и сглаживания представлены в таблице 1. Столбец «*Split*» соответствует T_{train}/T_{test} . В качестве оценки моделей будем использовать медиану *MAPE*, т. к. она более устойчива к выбросам в отличие от среднего. Прочерки в таблице означают отсутствие результатов или аномальные значения *MAPE*.

Таблица 1: Результаты тестирования.

| Сглаживание | Модель | Split | Медиана MAPE | | |
|------------------------------|--------------|----------|--------------|--------|-------|
| | | | Abilene | PeMSD7 | Totem |
| Логарифм | AutoReg | 5000/500 | 0.18 | 0.094 | 0.358 |
| | XGBoost | | 0.219 | 0.119 | 0.314 |
| | Holt-Winters | | 0.168 | 0.109 | 0.317 |
| | Prophet | | 0.319 | 0.119 | 0.341 |
| | LSTM | | 0.262 | - | - |
| | GCN-LSTM | 3000/50 | 0.146 | - | - |
| Бокс-Кокс | AutoReg | 5000/500 | 0.295 | 0.106 | 0.355 |
| | XGBoost | | 0.241 | - | 0.310 |
| | Holt-Winters | | 0.218 | 0.125 | 0.330 |
| | Prophet | | 0.515 | 0.125 | 0.35 |
| Скользящее среднее | AutoReg | 5000/500 | 0.232 | 0.219 | 0.468 |
| | XGBoost | | 0.278 | 0.210 | 0.395 |
| | Holt-Winters | | 0.272 | 0.217 | 0.386 |
| | Prophet | | 0.403 | 0.224 | 0.400 |
| Экспоненциальное сглаживание | AutoReg | 5000/500 | 0.170 | 0.119 | 0.380 |
| | XGBoost | | 0.222 | 0.127 | 0.330 |
| | Holt-Winters | | 0.217 | 0.123 | 0.553 |
| | Prophet | | 0.379 | 0.129 | 0.410 |
| SSA | AutoReg | 5000/500 | 0.220 | 0.109 | 0.380 |
| | XGBoost | | 0.197 | 0.117 | 0.332 |
| | Holt-Winters | | 0.195 | 0.140 | 0.534 |
| | Prophet | | 0.370 | 0.116 | 0.386 |

Как видно из таблицы 1, модели *AutoReg* и *XGBoost* превосходят другие алгоритмы прогнозирования, в то время как *логарифмирование*, *экспоненциальное сглаживание* и *SSA* являются лучшими методами сглаживания, среди рассмотренных. В следующих разделах мы объединим логарифм с двумя другими алгоритмами фильтрации, чтобы еще больше повысить точность.

Несмотря на то, что модель *Holt-Winters* имеет аналогичные результаты с *AutoReg*, она устойчива к любым сглаживающим препроцессорам, кроме логарифма. Это можно объяснить тем фактом, что модель сама по себе представляет собой комбинацию трех уравнений сглаживания.

Модель *Prophet* (расширение *ARIMA* с учетом тенденций и сезонных колебаний) не обеспечивает конкурентоспособной точности и, кроме того, является более дорогостоящей в вычислительном отношении по сравнению с другими моделями.

Мы решили не фокусировать внимание на нейросетевых подходах (например, *LSTM* и *GCN-LSTM*), хотя они и дают лучшие одношаговые прогнозы среди других алгоритмов. Соответственно, нам не удалось обучить их до конкурентоспособной точности без тонкой настройки гипер-параметров, индивидуальных для каждого набора данных.

4.3 Результаты валидации

4.3.1 Одномерные модели

В этом разделе мы предоставляем лучшие результаты валидации среди одномерных моделей для рассматриваемых наборов данных. Результаты всех вычислительных экспериментов представлены в Приложении Б.

Abilene

Таблица 2: Лучшие результаты валидации одномерных моделей для Abilene

| Модель | Сглаживание | Размер Train | MAPE | | MAE, Мб | |
|----------|-----------------------------------|--------------|---------|---------|---------|---------|
| | | | Среднее | Медиана | Среднее | Медиана |
| XGB | SSA + Логарифм | 1000 | 1.969 | 0.196 | 120.640 | 13.373 |
| LightGBM | SSA + Логарифм | 2000 | 1.169 | 0.178 | 120.486 | 11.633 |
| LightGBM | SSA + Логарифм | 3000 | 1.366 | 0.180 | 125.024 | 11.725 |
| LightGBM | SSA + Логарифм | 4000 | 0.929 | 0.176 | 126.300 | 11.556 |
| AutoReg | Экспоненциальное + Логарифм | 5000 | 0.672 | 0.172 | 124.519 | 12.186 |

Как видно из таблицы 2, на маленьких окнах наилучшие результаты дают модели, основанные на деревьях решений, но в то же время *AR* модель выигрывает на больших размерах обучающей выборки. *SSA* с логарифмированием лучше подходит для сглаживания, так как данные сильно зашумлены.

Totem

Таблица 3: Лучшие результаты валидации одномерных моделей для Totem

| Модель | Сглаживание | Размер Train | MAPE | | MAE, Мб | |
|----------|-------------------|--------------|---------|---------|---------|---------|
| | | | Среднее | Медиана | Среднее | Медиана |
| XGB | SSA + Логарифм | 1000 | 0.612 | 0.305 | 491.265 | 37.828 |
| LightGBM | SSA + Логарифм | 2000 | 0.652 | 0.292 | 91.328 | 31.977 |
| XGB | SSA + Логарифм | 3000 | 0.672 | 0.304 | 96.544 | 33.539 |
| LightGBM | SSA + Логарифм | 4000 | 0.678 | 0.316 | 102.732 | 31.959 |
| XGB | SSA + Логарифм | 5000 | 0.678 | 0.320 | 93.730 | 25.749 |

По результатам Таблицы 3 модели *LightGBM* и *XGB* дают наилучшую производительность для всех размеров обучающей выборки. Также, как и в случае с *Abilene*, *SSA* с логарифмированием лучше фильтруют данные от шума.

PeMSD7

Таблица 4: Лучшие результаты валидации одномерных моделей для PeMSD7

| Модель | Сглаживание | Размер Train | MAPE | | MAE | |
|----------|-------------|--------------|---------|---------|---------|---------|
| | | | Среднее | Медиана | Среднее | Медиана |
| LightGBM | Логарифм | 1000 | 0.126 | 0.105 | 5.231 | 4.686 |
| LightGBM | Логарифм | 2000 | 0.132 | 0.109 | 5.366 | 4.824 |
| AutoReg | Логарифм | 3000 | 0.119 | 0.100 | 5.168 | 4.617 |
| AutoReg | Логарифм | 4000 | 0.118 | 0.100 | 5.069 | 4.572 |
| AutoReg | Логарифм | 5000 | 0.118 | 0.098 | 5.042 | 4.500 |

Судя по таблице 4, модель *AutoReg* дает наилучшие результаты на больших окнах. В других случаях лучше использовать *LightGBM*. Для этого набора данных в качестве сглаживания лучше подходит обычный логарифм, так как *PeMSD7* не такой зашумленный, как остальные.

4.3.2 Многомерные модели

В этом разделе мы предоставляем лучшие результаты валидации среди многомерных моделей для рассматриваемых наборов данных. Результаты всех вычислительных экспериментов представлены в Приложении В.

Abilene

Таблица 5: Лучшие результаты валидации многомерных моделей для Abilene

| Модель | Сглаживание | Размер Train | MAPE | | MAE, Мб | |
|----------|-------------------|--------------|---------|---------|---------|---------|
| | | | Среднее | Медиана | Среднее | Медиана |
| XGB | Логарифм | 1000 | 1.654 | 0.167 | 115.462 | 10.947 |
| XGB | Логарифм | 2000 | 1.151 | 0.156 | 117.365 | 10.250 |
| LightGBM | Логарифм | 3000 | 1.702 | 0.162 | 120.723 | 10.267 |
| LightGBM | SSA + Логарифм | 4000 | 0.700 | 0.152 | 122.680 | 10.076 |
| LightGBM | Логарифм | 5000 | 0.568 | 0.155 | 124.871 | 9.940 |

Как видно из таблицы 5, модели *LightGBM* и *XGB* показывают лучшие результаты. При этом в большинстве случаев для фильтрации было достаточно одного логарифма. Наименьшие медианы *MAPE* и *MAE* достигаются на больших окнах размером 4000 и 5000.

Totem

Таблица 6: Лучшие результаты валидации многомерных моделей для Totem

| Модель | Сглаживание | Размер Train | MAPE | | MAE, Мб | |
|--------|-------------|--------------|---------|---------|---------|---------|
| | | | Среднее | Медиана | Среднее | Медиана |
| XGB | Логарифм | 1000 | 0.598 | 0.331 | 486.628 | 38.468 |
| XGB | Логарифм | 2000 | 0.647 | 0.323 | 90.845 | 33.843 |
| XGB | Логарифм | 3000 | 0.701 | 0.335 | 95.830 | 33.398 |
| XGB | Логарифм | 4000 | 0.645 | 0.356 | 101.365 | 33.476 |
| XGB | Логарифм | 5000 | 0.674 | 0.346 | 92.542 | 31.832 |

Как видно из таблицы 6, модели на основе деревьев решений снова оказались лучше авторегрессионных. В это время *XGB* показывает лучшие результаты на всех размерах обучающей выборки. И на удивление обычный логарифм оказался лучше более сложных методов сглаживания, несмотря на шум в данных.

PeMSD7

Таблица 7: Лучшие результаты валидации многомерных моделей для PeMSD7

| Модель | Сглаживание | Размер Train | MAPE | | MAE | |
|----------|-------------|--------------|---------|---------|---------|---------|
| | | | Среднее | Медиана | Среднее | Медиана |
| XGB | Логарифм | 1000 | 0.067 | 0.049 | 2.961 | 2.316 |
| LightGBM | Логарифм | 2000 | 0.063 | 0.046 | 2.732 | 2.150 |
| LightGBM | Логарифм | 3000 | 0.063 | 0.045 | 2.735 | 2.128 |
| LightGBM | Логарифм | 4000 | 0.062 | 0.044 | 2.682 | 2.065 |
| LightGBM | Логарифм | 5000 | 0.062 | 0.044 | 2.690 | 2.083 |

Мы получили следующие результаты (таблица 7). Практически во всех случаях лучшие показатели дает модель *LightGBM*. Более того, методы, основанные на деревьях решений, в несколько раз превзошли авторегрессионную модель. Поскольку данные были не сильно зашумлены, то для сглаживания оказалось достаточно одного логарифмирования. Если сравнивать полученные результаты с существующими современными подходами, такими как *GNN* [24], то наши модели близки к ним по точности, а возможно, даже лучше.

4.4 Анализ результатов

Мы адаптировали ряд алгоритмов прогнозирования временных рядов (*AR*, *VAR*, *Holt-Winters*, *XGBoost*, *LightGBM*) для прогнозирования трафика в сети. Также мы дополнительно проверили влияние сглаживания (*логарифмирование*, *преобразование Бокс-Кокса*, *скользящее среднее*, *экспоненциальное сглаживание*, *SSA*) на точность этих алгоритмов. Чтобы провести тщательную оценку качества прогноза, мы определили протокол проверки, основанный на скользящем окне. Для проверки наличия пространственных зависимостей и их использования, мы разработали простые многомерные версии всех использованных в исследовании алгоритмов одномерного прогнозирования (кроме модели *Holt-Winters*). Чтобы изучить чувствительность точности прогноза к длине обучающего временного ряда, мы протестировали алгоритмы на размерах окна в диапазоне от 1000 до 5000. На основании полученных результатов можно сделать следующие выводы:

- Логарифмическое преобразование значительно повышает точность прогноза.
- Модели, основанные на деревьях решений, обычно превосходят другие.
- Для интернет-трафика *AR*, самая простая статистическая модель, которую мы использовали, является конкурентоспособным алгоритмом в одномерном случае с дополнительной поддержкой сезонности.
- Увеличение окна обучения не обязательно повышает точность. Таким образом, меньший размер тренировочной выборки может быть предпочтительнее, поскольку он позволяет алгоритму прогнозирования быть адаптивным.
- Для интернет-трафика *SSA* превосходит экспоненциальное сглаживание и скользящее среднее.
- Многомерные алгоритмы превосходят одномерные, что подразумевает наличие пространственных зависимостей, которые мы не смогли выявить при корреляционном анализе данных.

- Наши лучшие результаты сравнимы с самыми современными решениями.

Таким образом, если задача состоит в простом прогнозировании временных рядов, следует использовать алгоритмы *XGBoost* или *LightGBM*. С другой стороны, хотя авторегрессионные методы уступают по точности, их можно использовать для вывода зависимостей между компонентами системы, что делает их полезными для реконструкции сети и задач вывода причинно-следственных связей. Примеры результатов прогнозирования моделей можно найти в Приложении Г (Рисунок 27 для статистических моделей и Рисунок 28 для моделей бустинга).

Заключение

Целью данного исследования была разработка эффективных алгоритмов для прогнозирования сетевого трафика. Созданная библиотека содержит методы как для сглаживания временных рядов, так и для их предсказания и валидации построенных моделей (одномерных или многомерных). Были получены следующие результаты: на примере набора данных *PeMSD7* в одномерном случае лучшие показатели, а именно $MAPE = 0.098$, $MAE = 4.5$, были достигнуты авторегрессионной моделью с логарифмическим преобразованием данных, а среди многомерных алгоритмов – *LightGBM* с логарифмированием ($MAPE = 0.044$, $MAE = 2.065$). Реализованные многомерные методы прогнозирования превосходят одномерные аналоги, а также они способны конкурировать с современными нейросетевыми подходами и при некоторых условиях даже превосходить их. Например, точность предсказания модели *LightGBM* с использованием логарифмирования на наборе данных *PeMSD7* сравнима с моделью, основанной на графовых нейронных сетях, из работы [24].

Точность моделей прогнозирования напрямую зависит от данных, их зашумленности, а также от размера обучающей выборки. Результаты работы алгоритмов улучшались с использованием различных способов фильтрации, особенно их комбинаций.

В ходе работы были выполнены следующие задачи:

1. изучены возможные решения данной проблемы;
2. найдены и проанализированы открытые наборы данных, имеющие сетевую структуру;
3. реализованы все необходимые алгоритмы прогнозирования, сглаживания и валидации;
4. проведено обучение, тестирование и валидация моделей;
5. проанализированы полученные результаты, выполнено сравнение реализованных моделей и сделаны соответствующие выводы.

Подводя итог, можно отметить, что задача прогнозирования трафика сетевой системы является довольно сложной и требует не только разработки алгоритмов предсказания, но и тщательного анализа данных, поиска пространственных зависимостей элементов сети, адаптации моделей к смене динамики, устойчивости к выбросам.

Список литературы

- [1] Alasmar, M., Parisi, G., Clegg, R., and Zakhleniu, N. On the distribution of traffic volumes in the internet and its implications. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications* (2019), IEEE, pp. 955–963.
- [2] Alrumaih, R. M., and Al-Fawzan, M. A. Time series forecasting using wavelet denoising an application to saudi stock index. *Journal of King Saud University - Engineering Sciences* 14, 2 (2002), 221–233.
- [3] Brockwell, P. J., and Davis, R. A. *Introduction to time series and forecasting*. Springer, 2002.
- [4] Chen, X., Wang, H., Wei, Y., Li, J., and Gao, H. Autoregressive-model-based methods for online time series prediction with missing values: an experimental evaluation. *ArXiv abs/1908.06729* (2019).
- [5] Chen, Y.-y., Yisheng, L., and Li, Z. Long short-term memory model for traffic congestion prediction with online open data. pp. 132–137.
- [6] Cortez, P., Rio, M., Rocha, M., and Sousa, P. Multi-scale internet traffic forecasting using neural networks and time series methods. *Expert Systems* 29 (05 2012), 143–155.
- [7] Deineko, Z. Wavelet coherence as a tool for visualization of complex physical processes.
- [8] Direction Generale des Technologies, d. l. R. e. d. l. o. t. W. g. Totem dataset, 2008.
- [9] Du, Y., Wang, J., Feng, W., Pan, S. J., Qin, T., Xu, R., and Wang, C. Adarnn: Adaptive learning and forecasting of time series. *Proceedings of the 30th ACM International Conference on Information & Knowledge Management* (2021).
- [10] Friedman, J. H. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics* 29, 5 (2001), 1189–1232.

- [11] Jiang, W., and Luo, J. Graph neural network for traffic forecasting: A survey. *Expert Systems with Applications* 207 (2022), 117921.
- [12] Lu, Z., Lv, W., Xie, Z., Du, B., and Huang, R. Leveraging graph neural network with lstm for traffic speed prediction. In *2019 IEEE SmartWorld, Ubiquitous Intelligence and Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)* (2019), pp. 74–81.
- [13] Lütkepohl, H. *New introduction to multiple time series analysis*. Springer Science & Business Media, 2005.
- [14] Makridakis, S., Spiliotis, E., and Assimakopoulos, V. The m4 competition: Results, findings, conclusion and way forward. *International Journal of Forecasting* 34, 4 (2018), 802–808.
- [15] Makridakis, S., Spiliotis, E., and Assimakopoulos, V. M5 accuracy competition: Results, findings, and conclusions. *International Journal of Forecasting* (2022).
- [16] Mehmood, H., Kostakos, P., Cortes, M., Anagnostopoulos, T., Pirttikangas, S., and Gilman, E. Concept drift adaptation techniques in distributed environment for real-world data streams. *Smart Cities* 4, 1 (2021), 349–371.
- [17] Ming, Z., Zhang, L., Wu, H., Xu, Y., Bakshi, M., Bai, B., and Zhang, G. A convergent semi-proximal alternating direction method of multipliers for recovering internet traffics from link measurements, 02 2021.
- [18] Pope, S. B., and Pope, S. B. *Turbulent flows*. Cambridge university press, 2000.
- [19] Tikunov, D., and Nishimura, T. Traffic prediction for mobile network using holt-winters’s exponential smoothing. *2007 15th International Conference on Software, Telecommunications and Computer Networks* (2007), 1–5.

- [20] Troia, S., Alvizu, R., Zhou, Y., Maier, G., and Pattavina, A. Deep learning-based traffic prediction for network optimization. In *2018 20th International Conference on Transparent Optical Networks (ICTON)* (2018), pp. 1–4.
- [21] van der Heijden, M., Velikova, M., and Lucas, P. J. Learning bayesian networks for clinical time series analysis. *Journal of Biomedical Informatics* 48 (2014), 94–105.
- [22] VeritasYin. Stgcn_ijcai-18, pemsd7 dataset, 2019.
- [23] Yang, H., Pan, Z., Tao, Q., and Qiu, J. Online learning for vector autoregressive moving-average time series prediction. *Neurocomputing* 315 (2018), 9–17.
- [24] Yu, B., Yin, H., and Zhu, Z. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. pp. 3634–3640.
- [25] Zhang, Y. Abilene dataset, 2014.
- [26] Zhao, J., Zhu, T., Zhao, R., and Zhao, P. *Layerwise Recurrent Autoencoder for Real-World Traffic Flow Forecasting*. 11 2019, pp. 78–88.

Приложение А. Обзор литературы

Существующие методы решения поставленной задачи

Таблица 8: Существующие решения задачи прогнозирования сетевого трафика

| Статья | Набор данных | Методы | Горизонт прогнозирования | Валидация | Результаты |
|--|--------------------------------------|---|--------------------------|--------------|--|
| Deep Learning-based Traffic Prediction for Network Optimization [20] | Abilene | Gated Recurrent Units | 1 час (12 шагов) | Тестирование | $Mae \leq 7.4$ |
| Traffic Prediction for Mobile Network using Holt-Winter's Exponential Smoothing [19] | Частные GSM, GPRS сети | Экспоненциальное сглаживание Холта-Винтера | Многошаговая | Тестирование | $0.1 \leq Nrmse \leq 0.3$ |
| Multi-scale Internet traffic forecasting using neural networks and time series methods [6] | SNMP (стандартный интернет-протокол) | Экспоненциальное сглаживание, ARIMA, ансамбли нейронных сетей | Одно- и многошаговая | Тестирование | Одношаговые: $1 < Mape < 7$. Многошаговые: $3 < Mape < 22$. |
| LSTM for Traffic Congestion Prediction with Online Open Data [5] | Google карты, AMAP | LSTM | Классификация | Тестирование | Precision = 0.98 Recall = 0.99 $F = 0.98$ |

| Продолжение таблицы | | | | | |
|---|--|--------------------------------------|--------------------------|---|---|
| Статья | Набор данных | Методы | Горизонт прогнозирования | Валидация | Результаты |
| Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting [24] | BJER4 и PeMSD7 | Graph Convolutional Networks | Много-шаговая | Тести-рование | $5 < Mape < 13$ |
| Leveraging Graph Neural Network with LSTM For Traffic Speed Prediction [12] | Данные о дорогах из района Бейлинь провинции Шэньси в Китае. | Graph long short term memory (GLSTM) | Одно-шаговая | Тести-рование | $Mae = 5$, $Mape = 0.36$, $Rmse = 6.86$ |
| Graph Neural Network for Traffic Forecasting: A Survey [11] | Большой набор открытых данных для транспортных задач | GNN, RNN в большинстве случаев | Разные варианты | + | $1.74 < Mae < 22$ $2.8 < Rmse < 36$ $3.7 < Mape < 16$ |
| Layerwise Recurrent Autoencoder for General Real-world Traffic Flow Forecasting [26] | ST-WB, ENG-HW | Autoencoder (SAO) + LSTM + GCN (CNN) | Одно- и много-шаговые | Тести-рование | $6 < Mape < 16$ |
| A Convergent Semi-Proximal Alternating Direction Method of Multipliers for Recovering Internet Traffics from Link Measurements [17] | Abilene, Totem GEANT, HOD | Sparsity Low-Rank Recovery Model | Много-шаговая | Перекрестная валидация, валидация Монте-Карло | Abilene: $0.05 < Nmae < 0.2$ Totem: $0.1 < Nmae < 0.8$ HOD: $0.15 < Nmae < 0.16$ |

Методы онлайн обучения и адаптации

Таблица 9: Обзор методов онлайн-адаптации и онлайн-обучения

| Статья | Метод адаптации | Модель | Комментарии |
|---|---|---|--|
| Online learning for vector autoregressive moving-average time series prediction [23] | Online Gradient Descent (OGD), Online Newton Step (ONS) | VAR, VARMA | Предлагаемый алгоритм VARMA-ONS действителен только для экспоненциально-вогнутой функции потерь, а алгоритм VARMA-OGD – для общей выпуклой функции потерь. |
| AdaRNN: Adaptive Learning and Forecasting for Time Series [9] | Адаптивные RNNs | Адаптивные RNNs | AdaRNN состоит из двух новых алгоритмов: характеристики временного распределения для получения информации о распределении в временного ряда и сопоставления временного распределения для построения обобщенной модели RNN путем сопоставления распределения. |
| Concept Drift Adaptation Techniques in Distributed Environment for Real-World Data Streams [16] | Тест Пейджа Хинкли; Адаптивный оконный режим; Метод обнаружения дрейфа; Метод раннего обнаружения дрейфа | Тригонометрическое преобразование Бокса-Кокса; ARMA; Prophet; LSTM | В этой статье представлены реализация и анализ выбранных современных методов обнаружения дрейфа концепции для анализа временных рядов в распределенной среде. |

| Продолжение таблицы | | | |
|--|---|--------------------------|--|
| Статья | Метод адаптации | Модель | Комментарии |
| Autoregressive-Model-Based Methods for Online Time Series Prediction with Missing Values: an Experimental Evaluation [4] | Оценщик Юла-Уокера (YW); Фильтр Калмана (KF); Онлайн градиентный спуск (OGD); Эффективная регрессия Ridge (AERR) | Авторегрессионные модели | AERR имеет наихудшую производительность, и она особенно слаба на временных рядах, где диапазон вариаций велик, изменение значения наблюдения резкое и частота пропуска высока. Несмотря на лучшую производительность по сравнению с AERR, OGD по-прежнему имеет низкую производительность, а также чувствителен к пропущенным значениям в реальных данных. KF и YW в целом имеют лучшую производительность. Однако метод YW не подходит для временных рядов без сезонности или с неустойчивым трендом, а KF менее эффективен в случае, если шум не является гауссовым. |

Приложение Б. Результаты валидации одномерных моделей для трафика по узлам

Abilene

Таблица 10: Результаты валидации одномерных моделей для Abilene

| Модель | Сглаживание | Размер окна | MAPE | | MAE, Мб | |
|--------|---|-------------|---------|---------|---------|---------|
| | | | Среднее | Медиана | Среднее | Медиана |
| AR | Логарифмирование | 1000 | 1.308 | 0.248 | 127.873 | 16.202 |
| | | 2000 | 1.663 | 0.211 | 125.887 | 14.223 |
| | | 3000 | 1.511 | 0.186 | 122.391 | 12.467 |
| | | 4000 | 0.702 | 0.183 | 122.566 | 12.376 |
| | | 5000 | 0.674 | 0.175 | 124.084 | 12.231 |
| | Экспоненциальное + Логарифмирование | 1000 | 1.360 | 0.250 | 127.011 | 16.197 |
| | | 2000 | 2.030 | 0.213 | 148.199 | 14.697 |
| | | 3000 | 1.461 | 0.183 | 121.861 | 12.332 |
| | | 4000 | 0.688 | 0.182 | 122.845 | 12.204 |
| | | 5000 | 0.672 | 0.172 | 124.519 | 12.186 |
| | SSA + Логарифмирование | 1000 | 1.314 | 0.250 | 134.338 | 16.063 |
| | | 2000 | 2.301 | 0.227 | 160.622 | 16.054 |
| | | 3000 | 2.205 | 0.201 | 129.921 | 13.413 |
| | | 4000 | 0.694 | 0.198 | 128.833 | 13.626 |
| | | 5000 | 0.664 | 0.191 | 124.638 | 13.230 |
| XGB | Логарифмирование | 1000 | 2.948 | 0.201 | 266.551 | 13.870 |
| | | 2000 | 1.601 | 0.186 | 191.969 | 12.503 |
| | | 3000 | 1.803 | 0.191 | 195.311 | 12.175 |
| | | 4000 | 1.106 | 0.184 | 156.752 | 12.198 |
| | | 5000 | 0.989 | 0.182 | 153.977 | 11.908 |
| | Экспоненциальное + Логарифмирование | 1000 | - | 0.233 | - | 16.741 |
| | | 2000 | - | 0.218 | - | 15.365 |
| | | 3000 | - | 0.221 | - | 15.631 |
| | | 4000 | - | 0.215 | - | 15.449 |
| | | 5000 | - | 0.214 | - | 14.764 |

| Продолжение таблицы | | | | | | |
|---------------------|---|-------------|---------|---------|---------|---------|
| Модель | Сглаживание | Размер окна | MAPE | | MAE, Мб | |
| | | | Среднее | Медиана | Среднее | Медиана |
| XGB | SSA + Логарифмирование | 1000 | 1.969 | 0.196 | 120.640 | 13.373 |
| | | 2000 | 1.165 | 0.179 | 119.928 | 11.845 |
| | | 3000 | 1.264 | 0.181 | 123.501 | 11.776 |
| | | 4000 | 0.905 | 0.179 | 125.181 | 11.586 |
| | | 5000 | 0.863 | 0.178 | 126.427 | 11.451 |
| LightGBM | Логарифмирование | 1000 | 2.192 | 0.215 | 137.513 | 14.572 |
| | | 2000 | 1.383 | 0.184 | 128.956 | 11.848 |
| | | 3000 | 1.492 | 0.183 | 131.252 | 11.999 |
| | | 4000 | 0.988 | 0.179 | 132.627 | 11.685 |
| | | 5000 | 0.921 | 0.181 | 130.954 | 11.737 |
| | Экспоненциальное + Логарифмирование | 1000 | - | 0.243 | - | 17.225 |
| | | 2000 | - | 0.217 | - | 14.770 |
| | | 3000 | - | 0.215 | - | 15.071 |
| | | 4000 | - | 0.209 | - | 14.293 |
| | | 5000 | - | 0.209 | - | 13.987 |
| | SSA + Логарифмирование | 1000 | 1.951 | 0.210 | 121.561 | 13.960 |
| | | 2000 | 1.169 | 0.178 | 120.486 | 11.633 |
| | | 3000 | 1.366 | 0.180 | 125.024 | 11.725 |
| | | 4000 | 0.929 | 0.176 | 126.300 | 11.556 |
| | | 5000 | 0.896 | 0.175 | 126.578 | 11.444 |

Таблица 11: Результаты валидации одномерных моделей для PeMSD7

| Модель | Сглаживание | Размер окна | MAPE | | MAE, Мб | |
|--------|---|-------------|---------|---------|---------|---------|
| | | | Среднее | Медиана | Среднее | Медиана |
| AR | Логарифмирование | 1000 | 0.148 | 0.119 | 6.738 | 5.509 |
| | | 2000 | 0.131 | 0.109 | 5.806 | 5.057 |
| | | 3000 | 0.119 | 0.100 | 5.168 | 4.617 |
| | | 4000 | 0.118 | 0.100 | 5.069 | 4.572 |
| | | 5000 | 0.118 | 0.098 | 5.042 | 4.500 |
| | Экспоненциальное + Логарифмирование | 1000 | 0.150 | 0.119 | 6.885 | 5.527 |
| | | 2000 | 0.133 | 0.110 | 5.913 | 5.104 |
| | | 3000 | 0.120 | 0.100 | 5.176 | 4.621 |
| | | 4000 | 0.118 | 0.100 | 5.074 | 4.569 |
| | | 5000 | 0.118 | 0.098 | 5.043 | 4.506 |
| | SSA + Логарифмирование | 1000 | 0.168 | 0.131 | 7.758 | 6.087 |
| | | 2000 | 0.153 | 0.121 | 6.862 | 5.583 |
| | | 3000 | 0.138 | 0.113 | 6.112 | 5.218 |
| | | 4000 | 0.135 | 0.112 | 5.864 | 5.128 |
| | | 5000 | 0.134 | 0.113 | 5.845 | 5.113 |
| XGB | Логарифмирование | 1000 | 0.128 | 0.106 | 5.274 | 4.707 |
| | | 2000 | 0.132 | 0.110 | 5.426 | 4.843 |
| | | 3000 | 0.143 | 0.117 | 5.886 | 5.274 |
| | | 4000 | 0.143 | 0.118 | 5.878 | 5.255 |
| | | 5000 | 0.146 | 0.118 | 5.990 | 5.387 |
| | Экспоненциальное + Логарифмирование | 1000 | 0.315 | 0.123 | 17.070 | 5.576 |
| | | 2000 | 0.855 | 0.131 | 34.349 | 5.986 |
| | | 3000 | 0.531 | 0.144 | 30.993 | 6.607 |
| | | 4000 | 2.173 | 0.145 | 138.727 | 6.670 |
| | | 5000 | 0.882 | 0.149 | 53.012 | 6.938 |

| Продолжение таблицы | | | | | | |
|---------------------|---|-------------|---------|---------|---------|---------|
| Модель | Сглаживание | Размер окна | MAPE | | MAE, Мб | |
| | | | Среднее | Медиана | Среднее | Медиана |
| XGB | SSA + Логарифмирование | 1000 | 0.132 | 0.108 | 5.510 | 4.809 |
| | | 2000 | 0.133 | 0.110 | 5.480 | 4.836 |
| | | 3000 | 0.140 | 0.115 | 5.813 | 5.170 |
| | | 4000 | 0.141 | 0.116 | 5.809 | 5.161 |
| | | 5000 | 0.144 | 0.118 | 5.931 | 5.277 |
| LightGBM | Логарифмирование | 1000 | 0.126 | 0.105 | 5.231 | 4.686 |
| | | 2000 | 0.132 | 0.109 | 5.366 | 4.824 |
| | | 3000 | 0.139 | 0.114 | 5.651 | 5.113 |
| | | 4000 | 0.139 | 0.114 | 5.624 | 5.083 |
| | | 5000 | 0.140 | 0.115 | 5.682 | 5.138 |
| | Экспоненциальное + Логарифмирование | 1000 | 2.380 | 0.121 | 156.259 | 5.597 |
| | | 2000 | 4.415 | 0.129 | 292.412 | 5.900 |
| | | 3000 | 3.068 | 0.137 | 199.984 | 6.322 |
| | | 4000 | 2.576 | 0.137 | 165.733 | 6.246 |
| | | 5000 | 3.141 | 0.139 | 201.441 | 6.376 |
| | SSA + Логарифмирование | 1000 | 0.129 | 0.106 | 5.389 | 4.758 |
| | | 2000 | 0.133 | 0.110 | 5.407 | 4.795 |
| | | 3000 | 0.138 | 0.114 | 5.638 | 5.070 |
| | | 4000 | 0.139 | 0.114 | 5.633 | 5.055 |
| | | 5000 | 0.141 | 0.115 | 5.702 | 5.133 |

Totem

Таблица 12: Результаты валидации одномерных моделей для Totem

| Модель | Сглаживание | Размер окна | MAPE | | MAE, Мб | |
|--------|---|-------------|---------|---------|---------|---------|
| | | | Среднее | Медиана | Среднее | Медиана |
| AR | Логарифмирование | 1000 | 0.871 | 0.399 | 532.454 | 47.437 |
| | | 2000 | 0.795 | 0.392 | 123.067 | 43.047 |
| | | 3000 | - | 0.402 | - | 43.290 |
| | | 4000 | - | 0.447 | - | 46.913 |
| | | 5000 | - | 0.436 | - | 45.350 |
| | Экспоненциальное + Логарифмирование | 1000 | 0.968 | 0.392 | 540.955 | 48.603 |
| | | 2000 | 0.937 | 0.390 | 146.230 | 42.378 |
| | | 3000 | - | 0.399 | - | 42.721 |
| | | 4000 | - | 0.443 | - | 44.869 |
| | | 5000 | - | 0.435 | - | 45.380 |
| | SSA + Логарифмирование | 1000 | - | 0.438 | - | 56.657 |
| | | 2000 | - | 0.415 | - | 46.864 |
| | | 3000 | 4.379 | 0.411 | 281.328 | 49.486 |
| | | 4000 | 1.717 | 0.438 | 205.225 | 50.634 |
| | | 5000 | 3.942 | 0.421 | 215.272 | 45.911 |
| XGB | Логарифмирование | 1000 | 6.185 | 0.381 | - | 44.176 |
| | | 2000 | 3.496 | 0.408 | - | 41.899 |
| | | 3000 | 0.717 | 0.380 | 111.244 | 33.197 |
| | | 4000 | 0.697 | 0.390 | 119.558 | 32.965 |
| | | 5000 | 0.721 | 0.410 | 115.831 | 31.954 |
| | Экспоненциальное + Логарифмирование | 1000 | - | 0.549 | - | 71.704 |
| | | 2000 | - | 0.554 | - | 62.522 |
| | | 3000 | - | 0.528 | - | 58.668 |
| | | 4000 | - | 0.620 | - | 67.607 |
| | | 5000 | - | 0.692 | - | 72.645 |

| Продолжение таблицы | | | | | | |
|---------------------|---|-------------|---------|---------|---------|---------|
| Модель | Сглаживание | Размер окна | MAPE | | MAE, Мб | |
| | | | Среднее | Медиана | Среднее | Медиана |
| XGB | SSA + Логарифмирование | 1000 | 0.612 | 0.305 | 491.265 | 37.828 |
| | | 2000 | 0.640 | 0.303 | 91.819 | 33.881 |
| | | 3000 | 0.672 | 0.304 | 96.544 | 33.539 |
| | | 4000 | 0.672 | 0.330 | 101.335 | 30.099 |
| | | 5000 | 0.678 | 0.320 | 93.730 | 25.749 |
| LightGBM | Логарифмирование | 1000 | 1.518 | 0.383 | 800.389 | 45.242 |
| | | 2000 | 1.265 | 0.400 | 317.665 | 38.610 |
| | | 3000 | 0.702 | 0.364 | 114.791 | 33.076 |
| | | 4000 | 0.710 | 0.381 | 127.439 | 33.300 |
| | | 5000 | 0.733 | 0.396 | 126.771 | 29.777 |
| | Экспоненциальное + Логарифмирование | 1000 | - | 0.491 | - | 73.911 |
| | | 2000 | - | 0.478 | - | 58.513 |
| | | 3000 | - | 0.452 | - | 50.724 |
| | | 4000 | - | 0.473 | - | 54.142 |
| | | 5000 | - | 0.495 | - | 59.174 |
| | SSA + Логарифмирование | 1000 | 0.633 | 0.305 | 492.757 | 38.028 |
| | | 2000 | 0.652 | 0.292 | 91.328 | 31.977 |
| | | 3000 | 0.681 | 0.309 | 96.619 | 32.388 |
| | | 4000 | 0.678 | 0.316 | 102.732 | 31.959 |
| | | 5000 | 0.684 | 0.320 | 95.654 | 27.899 |

Приложение В. Результаты валидации многомерных моделей для трафика по узлам

Abilene

Таблица 13: Результаты валидации одномерных моделей для Abilene

| Модель | Сглаживание | Размер окна | MAPE | | MAE, Мб | |
|--------|---|-------------|---------|---------|---------|---------|
| | | | Среднее | Медиана | Среднее | Медиана |
| VAR | Логарифмирование | 1000 | 2.289 | 0.206 | 340.489 | 13.948 |
| | | 2000 | 1.692 | 0.174 | 183.088 | 12.134 |
| | | 3000 | 1.543 | 0.179 | 120.467 | 12.308 |
| | | 4000 | 0.817 | 0.173 | 121.048 | 12.201 |
| | | 5000 | 0.753 | 0.174 | 123.169 | 12.367 |
| | Экспоненциальное + Логарифмирование | 1000 | - | 0.192 | - | 12.211 |
| | | 2000 | - | 0.156 | - | 10.630 |
| | | 3000 | 0.740 | 0.158 | 23.107 | 10.734 |
| | | 4000 | 0.237 | 0.153 | 20.780 | 10.266 |
| | | 5000 | 0.222 | 0.153 | 20.618 | 10.383 |
| | SSA + Логарифмирование | 1000 | - | 0.190 | - | 12.370 |
| | | 2000 | 0.754 | 0.155 | 29.577 | 10.357 |
| | | 3000 | 0.770 | 0.154 | 22.620 | 10.475 |
| | | 4000 | 0.214 | 0.148 | 21.298 | 9.954 |
| | | 5000 | 0.208 | 0.148 | 19.949 | 10.008 |
| XGB | Логарифмирование | 1000 | 1.655 | 0.167 | 115.540 | 10.953 |
| | | 2000 | 1.151 | 0.156 | 117.370 | 10.259 |
| | | 3000 | 1.805 | 0.162 | 120.724 | 10.486 |
| | | 4000 | 0.705 | 0.158 | 122.638 | 10.249 |
| | | 5000 | 0.772 | 0.159 | 125.404 | 10.422 |
| | Экспоненциальное + Логарифмирование | 1000 | 0.998 | 0.133 | 20.592 | 8.618 |
| | | 2000 | 0.597 | 0.126 | 19.808 | 8.150 |
| | | 3000 | 0.848 | 0.127 | 20.684 | 8.179 |
| | | 4000 | 0.208 | 0.124 | 19.984 | 7.966 |
| | | 5000 | 0.214 | 0.123 | 20.588 | 7.939 |

| Продолжение таблицы | | | | | | |
|---------------------|---|-------------|---------|---------|---------|---------|
| Модель | Сглаживание | Размер окна | MAPE | | MAE, Мб | |
| | | | Среднее | Медиана | Среднее | Медиана |
| XGB | SSA + Логарифмирование | 1000 | 0.869 | 0.132 | 20.449 | 8.549 |
| | | 2000 | 0.623 | 0.122 | 19.523 | 7.974 |
| | | 3000 | 0.862 | 0.123 | 20.822 | 7.922 |
| | | 4000 | 0.202 | 0.122 | 19.962 | 7.999 |
| | | 5000 | 0.202 | 0.122 | 20.631 | 7.629 |
| LightGBM | Логарифмирование | 1000 | 1.822 | 0.171 | 116.076 | 11.174 |
| | | 2000 | 1.183 | 0.156 | 118.187 | 10.166 |
| | | 3000 | 1.702 | 0.162 | 120.723 | 10.267 |
| | | 4000 | 0.686 | 0.153 | 123.214 | 9.985 |
| | | 5000 | 0.568 | 0.155 | 124.871 | 9.940 |
| | Экспоненциальное + Логарифмирование | 1000 | 1.046 | 0.138 | 21.354 | 8.908 |
| | | 2000 | 0.620 | 0.127 | 21.244 | 8.253 |
| | | 3000 | 0.934 | 0.126 | 21.792 | 8.014 |
| | | 4000 | 0.224 | 0.123 | 21.392 | 7.940 |
| | | 5000 | 0.206 | 0.119 | 20.456 | 7.659 |
| | SSA + Логарифмирование | 1000 | 1.011 | 0.139 | 21.190 | 8.872 |
| | | 2000 | 0.650 | 0.125 | 20.920 | 8.314 |
| | | 3000 | 1.011 | 0.124 | 21.596 | 8.073 |
| | | 4000 | 0.209 | 0.121 | 20.884 | 7.752 |
| | | 5000 | 0.217 | 0.116 | 21.061 | 7.645 |

Таблица 14: Результаты валидации одномерных моделей для PeMSD7

| Модель | Сглаживание | Размер окна | MAPE | | MAE, Мб | |
|--------|---|-------------|---------|---------|---------|---------|
| | | | Среднее | Медиана | Среднее | Медиана |
| VAR | Логарифмирование | 1000 | - | 0.099 | - | 4.396 |
| | | 2000 | 0.110 | 0.092 | 4.501 | 3.998 |
| | | 3000 | 0.106 | 0.089 | 4.312 | 3.886 |
| | | 4000 | 0.107 | 0.089 | 4.330 | 3.902 |
| | | 5000 | 0.108 | 0.090 | 4.363 | 3.923 |
| | Экспоненциальное + Логарифмирование | 1000 | - | 0.080 | - | 3.741 |
| | | 2000 | - | 0.070 | - | 3.287 |
| | | 3000 | 0.083 | 0.067 | 3.651 | 3.145 |
| | | 4000 | 0.083 | 0.068 | 3.605 | 3.145 |
| | | 5000 | 0.084 | 0.069 | 3.639 | 3.192 |
| | SSA + Логарифмирование | 1000 | - | 0.099 | - | 4.827 |
| | | 2000 | - | 0.073 | - | 3.465 |
| | | 3000 | - | 0.068 | - | 3.240 |
| | | 4000 | 0.089 | 0.068 | 4.070 | 3.241 |
| | | 5000 | 0.086 | 0.069 | 3.930 | 3.313 |
| XGB | Логарифмирование | 1000 | 0.066 | 0.049 | 2.940 | 2.312 |
| | | 2000 | 0.063 | 0.046 | 2.749 | 2.141 |
| | | 3000 | 0.064 | 0.046 | 2.849 | 2.207 |
| | | 4000 | 0.062 | 0.045 | 2.773 | 2.136 |
| | | 5000 | 0.063 | 0.046 | 2.823 | 2.160 |
| | Экспоненциальное + Логарифмирование | 1000 | 0.049 | 0.034 | 2.256 | 1.684 |
| | | 2000 | 0.045 | 0.031 | 2.074 | 1.524 |
| | | 3000 | 0.045 | 0.031 | 2.111 | 1.550 |
| | | 4000 | 0.044 | 0.030 | 2.046 | 1.485 |
| | | 5000 | 0.044 | 0.030 | 2.078 | 1.499 |

| Продолжение таблицы | | | | | | |
|---------------------|---|-------------|---------|---------|---------|---------|
| Модель | Сглаживание | Размер окна | MAPE | | MAE, Мб | |
| | | | Среднее | Медиана | Среднее | Медиана |
| XGB | SSA + Логарифмирование | 1000 | 0.049 | 0.034 | 2.280 | 1.713 |
| | | 2000 | 0.045 | 0.030 | 2.087 | 1.529 |
| | | 3000 | 0.044 | 0.030 | 2.095 | 1.533 |
| | | 4000 | 0.043 | 0.029 | 2.031 | 1.482 |
| | | 5000 | 0.044 | 0.030 | 2.069 | 1.505 |
| LightGBM | Логарифмирование | 1000 | 0.067 | 0.050 | 2.871 | 2.287 |
| | | 2000 | 0.063 | 0.046 | 2.717 | 2.134 |
| | | 3000 | 0.063 | 0.045 | 2.736 | 2.116 |
| | | 4000 | 0.061 | 0.044 | 2.675 | 2.058 |
| | | 5000 | 0.061 | 0.044 | 2.672 | 2.072 |
| | Экспоненциальное + Логарифмирование | 1000 | 0.047 | 0.033 | 2.155 | 1.617 |
| | | 2000 | 0.044 | 0.030 | 2.010 | 1.473 |
| | | 3000 | 0.043 | 0.030 | 2.000 | 1.448 |
| | | 4000 | 0.043 | 0.029 | 1.952 | 1.412 |
| | | 5000 | 0.042 | 0.029 | 1.936 | 1.417 |
| | SSA + Логарифмирование | 1000 | 0.048 | 0.033 | 2.183 | 1.638 |
| | | 2000 | 0.044 | 0.030 | 2.025 | 1.473 |
| | | 3000 | 0.043 | 0.029 | 1.985 | 1.426 |
| | | 4000 | 0.042 | 0.028 | 1.949 | 1.397 |
| | | 5000 | 0.042 | 0.028 | 1.941 | 1.397 |

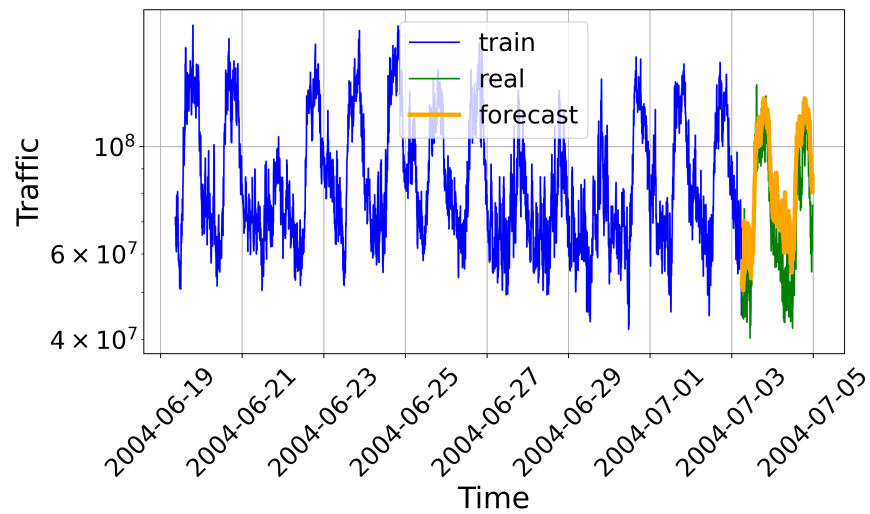
Totem

Таблица 15: Результаты валидации одномерных моделей для Totem

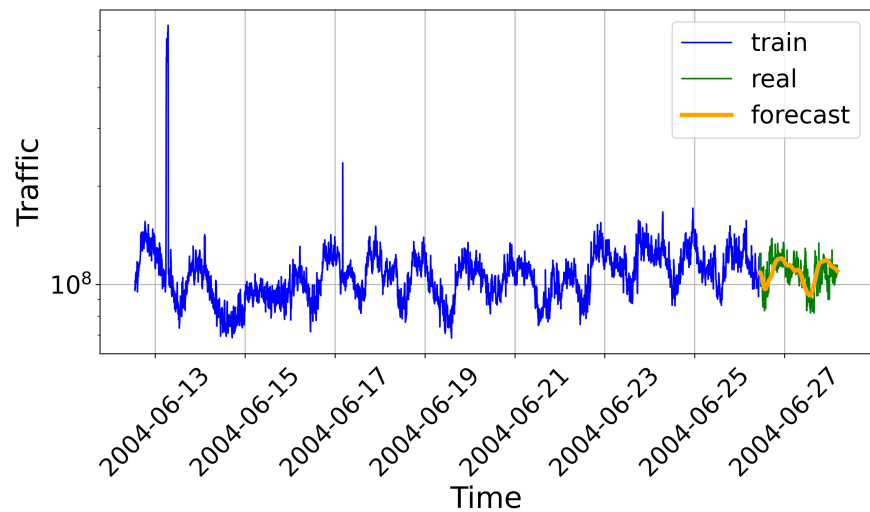
| Модель | Сглаживание | Размер окна | MAPE | | MAE, Мб | |
|--------|---|-------------|---------|---------|---------|---------|
| | | | Среднее | Медиана | Среднее | Медиана |
| VAR | Логарифмирование | 1000 | 0.622 | 0.384 | 497.684 | 44.497 |
| | | 2000 | 0.705 | 0.425 | 103.228 | 46.133 |
| | | 3000 | 0.830 | 0.477 | 109.861 | 47.474 |
| | | 4000 | 0.911 | 0.538 | 120.937 | 49.933 |
| | | 5000 | 0.958 | 0.546 | 116.683 | 47.952 |
| | Экспоненциальное + Логарифмирование | 1000 | 0.477 | 0.285 | 81.393 | 32.169 |
| | | 2000 | 0.533 | 0.317 | 82.134 | 34.154 |
| | | 3000 | 0.621 | 0.343 | 87.874 | 37.362 |
| | | 4000 | 0.666 | 0.380 | 96.965 | 39.204 |
| | | 5000 | 0.703 | 0.391 | 100.427 | 39.738 |
| | SSA + Логарифмирование | 1000 | - | 0.286 | - | 29.538 |
| | | 2000 | - | 0.296 | - | 31.541 |
| | | 3000 | 0.524 | 0.319 | 86.066 | 33.574 |
| | | 4000 | 0.509 | 0.342 | 93.627 | 36.232 |
| | | 5000 | 0.535 | 0.355 | 96.893 | 34.450 |
| XGB | Логарифмирование | 1000 | 0.589 | 0.311 | 484.856 | 34.948 |
| | | 2000 | 0.643 | 0.301 | 88.829 | 32.230 |
| | | 3000 | 0.664 | 0.327 | 94.357 | 33.994 |
| | | 4000 | 0.643 | 0.350 | 102.054 | 33.228 |
| | | 5000 | 0.746 | 0.348 | 94.350 | 30.905 |
| | Экспоненциальное + Логарифмирование | 1000 | 0.453 | 0.226 | 71.786 | 24.135 |
| | | 2000 | 0.486 | 0.225 | 70.229 | 22.259 |
| | | 3000 | 0.491 | 0.251 | 71.412 | 24.313 |
| | | 4000 | 0.523 | 0.269 | 81.343 | 25.131 |
| | | 5000 | 0.503 | 0.260 | 76.715 | 24.742 |

| Продолжение таблицы | | | | | | |
|---------------------|---|-------------|---------|---------|----------|---------|
| Модель | Сглаживание | Размер окна | MAPE | | MAE, Мб | |
| | | | Среднее | Медиана | Среднее | Медиана |
| XGB | SSA + Логарифмирование | 1000 | 0.458 | 0.225 | 70.204 | 23.953 |
| | | 2000 | 0.493 | 0.221 | 69.980 | 22.925 |
| | | 3000 | 0.493 | 0.236 | 71.667 | 24.260 |
| | | 4000 | 0.476 | 0.251 | 77.410 | 25.212 |
| | | 5000 | 0.519 | 0.263 | 76.985 | 25.092 |
| LightGBM | Логарифмирование | 1000 | 0.750 | 0.317 | 654.062 | 35.532 |
| | | 2000 | 0.805 | 0.352 | 277.382 | 36.776 |
| | | 3000 | 0.980 | 0.350 | 1969.199 | 37.112 |
| | | 4000 | 1.158 | 0.376 | 1456.911 | 34.910 |
| | | 5000 | 0.750 | 0.343 | 96.439 | 31.164 |
| | Экспоненциальное + Логарифмирование | 1000 | 0.456 | 0.230 | 72.586 | 24.360 |
| | | 2000 | 0.497 | 0.227 | 71.994 | 23.223 |
| | | 3000 | 0.496 | 0.245 | 72.527 | 24.552 |
| | | 4000 | 0.516 | 0.267 | 81.619 | 25.827 |
| | | 5000 | 0.551 | 0.260 | 79.456 | 24.409 |
| | SSA + Логарифмирование | 1000 | 0.489 | 0.227 | 72.729 | 25.297 |
| | | 2000 | 0.518 | 0.224 | 72.085 | 24.213 |
| | | 3000 | 0.500 | 0.235 | 71.579 | 24.235 |
| | | 4000 | 0.502 | 0.246 | 80.619 | 25.850 |
| | | 5000 | 0.532 | 0.249 | 79.004 | 24.195 |

Приложение Г. Визуализация результатов прогнозирования

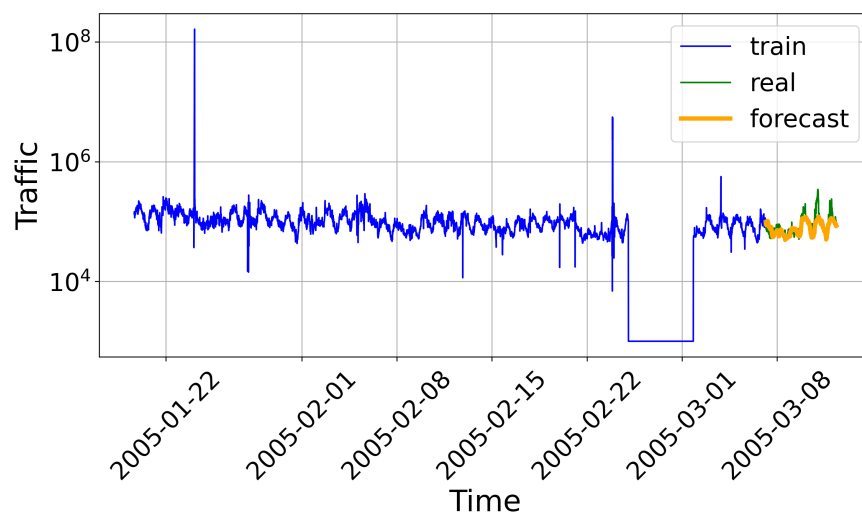


(a) Иллюстрация результатов прогнозирования авторегрессионной модели с логарифмированием для Abilene

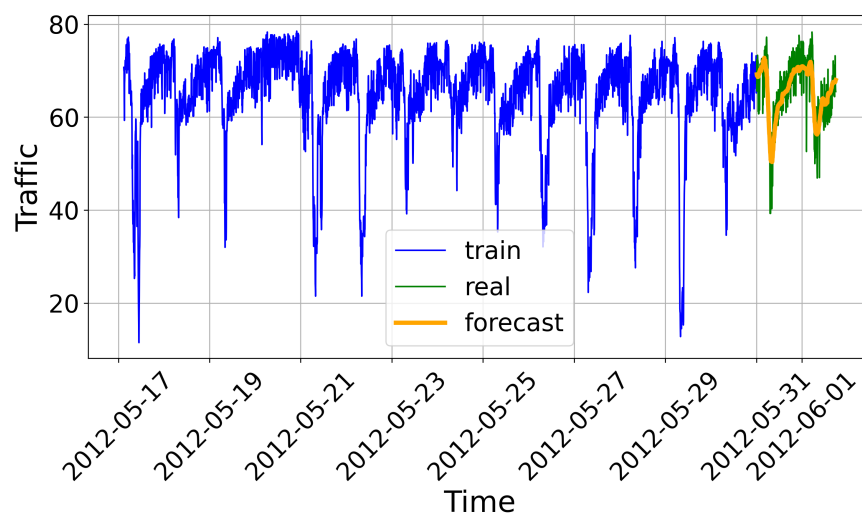


(b) Иллюстрация результатов прогнозирования VAR модели с SSA и логарифмированием для Abilene

Рис. 27: Иллюстрации результатов прогнозирования статистических моделей



(a) Иллюстрация результатов прогнозирования модели *XGBoost* с экспоненциальным сглаживанием и логарифмированием для Totem



(b) Иллюстрация результатов прогнозирования модели *LightGBM* с SSA и логарифмированием для PeMSD7

Рис. 28: Иллюстрации результатов прогнозирования моделей, основанных на деревьях решений