Федеральное государственное автономное образовательное учреждение
высшего образования
«Московский физико-технический институт
(национальный исследовательский университет)»
Физтех-школа Аэрокосмических Технологий
Институт аэромеханики и летательной техники

Направление подготовки / специальность: 09.03.01 Информатика и вычислительная техника
Направленность (профиль) подготовки: Компьютерное моделирование

АЛГОРИТМ ПРЕДИКТИВНОГО АНАЛИЗА ОТКАЗОВ СИСТЕМЫ ВИДЕОАНАЛИТИКИ В РЕЖИМЕ ВРЕМЕНИ ПО ДАННЫМ ОТ СИСТЕМ МОНИТОРИНГА

(бакалаврская работа)

Студент:
Боровец Николай Васильевич

_____
(подпись студента)

Научный руководитель:
Гришин Никита Александрович,

_____
(подпись научного руководителя)

Консультант *(при наличии)*:
_____
_____
(подпись консультанта)

Москва 2025

АННОТАЦИЯ

Настоящая работа посвящена проблеме обеспечения производительности современных систем видеоаналитики. Для предотвращения деградации качества обслуживания был разработан метод прогнозирования сквозной задержки обработки видеопотока.

В ходе исследования был проведен анализ многомерных временных рядов системных метрик, на основе которого были построены и сравнены прогностические модели, включая CatBoost и LSTM. Для оценки их качества и устойчивости применялась методология временной кросс-валидации.

Результатом работы стал прототип системы предиктивного мониторинга, способный с точностью MAPE 0.89% прогнозировать задержки на горизонте 3.75 часа. Разработанное решение является основой для создания интеллектуальной системы оповещений, которая позволяет принимать превентивные меры для поддержания стабильности видеоаналитического конвейера.

Ключевые слова: предиктивный анализ, временные ряды, мониторинг производительности, видеоаналитика, машинное обучение, LSTM, CatBoost, MLOps.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Обоснование выбора темы и актуальность

Современные системы видеоаналитики играют критически важную роль в обеспечении безопасности и мониторинга объектов критической инфраструктуры, включая аэродромы, железнодорожные станции, морские порты, промышленные предприятия и нефтеперерабатывающие комплексы [1]. Эти системы обрабатывают огромные объемы видеоданных в режиме реального времени, что предъявляет высокие требования к производительности и надежности всего технологического конвейера.

С ростом масштабов развертывания и усложнением архитектуры видеоаналитических систем возрастает и сложность их мониторинга [15]. Современные решения часто включают в себя многоуровневые конвейеры обработки, начиная от захвата видеопотоков с камер, их предварительной обработки, применения алгоритмов машинного обучения для детекции объектов и событий, передачи результатов через брокеры сообщений в бэкенд-системы и далее к конечным пользователям через веб-интерфейсы [12], [13], [16].

Повышение объемов данных и жестких требований к end-to-end задержкам (от момента возникновения события на видео до его отображения оператору) делает необходимым переход от реактивного к предиктивному подходу в управлении производительностью. Традиционные методы мониторинга, основанные на статических пороговых значениях и алертах по факту превышения SLA, не способны предотвратить деградацию качества обслуживания до ее критических проявлений [14].

В данном контексте особую важность приобретает разработка интеллектуальных систем предиктивного анализа, способных на основе потоковых метрик мониторинга (например, собираемых системой Prometheus [2] и визуализируемых в Grafana [3]) заблаговременно оценивать потенциальные проблемы производительности и инициировать превентивные меры по их устранению.

Цель и задачи исследования

Цель работы: разработать и внедрить комплексный метод предиктивного анализа задержек в конвейере видеоаналитики, способный прогнозировать конечную метрику $common\_event\_delay$ (также известную как $common\_cad$) с заданной точностью для предупреждения операторов о потенциальных сбоях до их фактического проявления.

Достижение поставленной цели требует решения комплекса взаимосвязанных *задач:*

1. Проведение аналитического обзора и систематизация современной литературы по предиктивному анализу временных рядов, машинному и глубокому обучению в контексте мониторинга и диагностики производительности систем реального времени [19].

2. Проведение анализа структуры и взаимных корреляций временных рядов метрик, собираемых на этапах видеоконвейера, включая выявление скрытых зависимостей между компонентами системы и идентификацию наиболее информативных признаков для прогнозирования.

3. Систематический обзор и сравнительный анализ современных методов прогнозирования временных рядов и обнаружения аномалий [18], включая классические статистические подходы, методы машинного обучения и глубокие нейронные сети, с оценкой их применимости к специфике видеоаналитических конвейеров.

4. Обоснованный выбор оптимальной архитектуры модели (градиентный бустинг, нейронные сети) с учетом требований к точности и скорости inference, а также определение необходимого объёма обучающих данных и оптимальной периодичности переобучения модели.

5. Проектирование и реализация полноценного MLOps-конвейера, включающего автоматизированный feature-engineering, механизмы периодического дообучения модели на новых данных, высокопроизводительный inference-сервис и системы мониторинга качества оценок.

6. Всестороннее экспериментальное исследование точности и производительности разработанной модели на обширных исторических данных с

использованием методов временной кросс-валидации и оценкой устойчивости к различным типам аномалий в данных.

7. Разработка и внедрение интеллектуальной системы оповещений с адаптивными порогами, а также формулирование практических рекомендаций по эксплуатации, настройке и масштабированию решения в производственной среде.

Методология и методы исследования

Для достижения поставленной цели и решения сформулированных задач применяется комплексная методология, сочетающая теоретические исследования с практическими экспериментами:

1. Организация непрерывного сбора и интеллектуальной предобработки потоковых метрик из системы мониторинга Prometheus [2], включая очистку от выбросов, нормализацию, обработку пропущенных значений и синхронизацию временных рядов различных компонентов системы.

2. Разработка специализированного модуля построения многомерных временных рядов с интеллектуальной генерацией признаков, включая временные лаги различной глубины, скользящие статистические агрегаты, спектральные характеристики и высокоразмерные эмбеддинги для захвата сложных временных зависимостей.

3. Реализация и экспериментальное сравнение различных архитектур моделей (градиентный бустинг, нейронные сети) с применением строгих методов перекрёстной валидации по времени для обеспечения корректной оценки обобщающей способности.

4. Контейнеризация решения с использованием технологии Docker и проведение детальных измерений latency inference в условиях, максимально приближенных к производственным, включая тестирование под нагрузкой и оценку масштабируемости.

Теоретическая и практическая значимость

Теоретическая значимость работы заключается в сравнительном анализе методов прогнозирования временных рядов для систем мониторинга и определении их применимости к задачам предиктивной диагностики в условиях жестких временных ограничений.

Практическая значимость определяется разработкой готового к промышленному использованию решения для мониторинга и предупреждения отказов видеоконвейера с гарантированным соблюдением SLA по конечной метрике $common\_event\_delay$. Созданная система может быть адаптирована и масштабирована для применения в различных отраслях, где критична надежность систем обработки потоковых данных в реальном времени, включая телекоммуникации, финансовые технологии и промышленную автоматизацию.

1 Общие положения

Данная глава посвящена формальной постановке задачи предиктивного анализа задержек в конвейере видеоаналитики и представлению архитектуры исследуемой системы. В рамках главы вводятся ключевые математические обозначения, определяются целевые метрики и ограничения, формулируются требования к разрабатываемому алгоритму. Особое внимание уделяется описанию структуры видеоконвейера и точек сбора телеметрических данных, которые лягут в основу построения прогностической модели.

1.1 Архитектура системы видеоаналитики

Исследуемая система видеоаналитики представляет собой многокомпонентный конвейер, предназначенный для обработки видеопотоков в режиме реального времени с применением алгоритмов компьютерного зрения для детекции событий и объектов. Архитектура системы строится по принципу микросервисной архитектуры, что обеспечивает масштабируемость и отказоустойчивость, но одновременно усложняет задачи мониторинга и диагностики производительности.

Видеоконвейер включает следующие основные компоненты: модуль захвата видеопотока с IP-камер (получающий данные по протоколу RTSP), ML-pipeline для применения алгоритмов компьютерного зрения, брокер сообщений Apache Kafka [4] для асинхронной передачи результатов обработки, бэкенд-сервисы для бизнес-логики и сохранения данных, а также WebSocket-клиенты для доставки уведомлений конечным пользователям. Каждый компонент генерирует множество метрик производительности, которые собираются централизованной системой мониторинга Prometheus [2].

Критической характеристикой системы является end-to-end-задержка, измеряемая как время от момента возникновения события в видеопотоке до его отображения на интерфейсе оператора. Данная метрика, обозначаемая как $common\_event\_delay$, напрямую влияет на эффективность работы опе-

раторов и качество принимаемых ими решений в критических ситуациях. Общая схема видеоконвейера и точки сбора метрик представлены на *рисунке 1.1*.

1.2 Постановка задачи

Для формальной постановки задачи прогнозирования введем необходимые математические обозначения и определения.

Дано

1. **Многомерный временной ряд** метрик, собираемых из системы мониторинга Prometheus с периодом дискретизации 15 секунд. В каждый момент времени t_i фиксируется d-мерный вектор наблюдений $\mathbf{x}_i = [m_i^{(1)}, \ldots, m_i^{(d)}] \in \mathbb{R}^d$, характеризующий состояние видеоконвейера. Объем доступных исторических данных составляет 90643 точки за 16 дней.

2. **Горизонт прогнозирования** $\Delta = 900$ временных шагов, что соответствует 3.75 часам.

Найти

1. **Целевую функцию** $f^* : \mathbb{R}^{L \times d} \to \mathbb{R}$, которая отображает историю наблюдений (представленную матрицей $X_k \in \mathbb{R}^{L \times d}$ из L последних векторов наблюдений) в будущее значение целевой метрики $y_k = common\_event\_delay(t_k + \Delta)$.

2. **Прогностический алгоритм** $A(X_k)$, который является наилучшей аппроксимацией целевой функции f^*.

Критерии качества и ограничения

Разрабатываемый алгоритм A должен удовлетворять следующим требованиям:

1. **Точность прогнозирования:** средняя абсолютная процентная ошибка (MAPE) на тестовых данных должна быть менее 10%.

2. **Производительность:** время вычисления одного прогноза (inference time) на целевом оборудовании не должно превышать 5 секунд.

3. **Интерпретируемость:** модель должна предоставлять возможность оценить важность признаков, влияющих на прогноз.

4. **Практичность:** решение должно быть реализовано в виде, пригодном для интеграции в существующий MLOps-конвейер, включая контейнеризацию с помощью Docker.

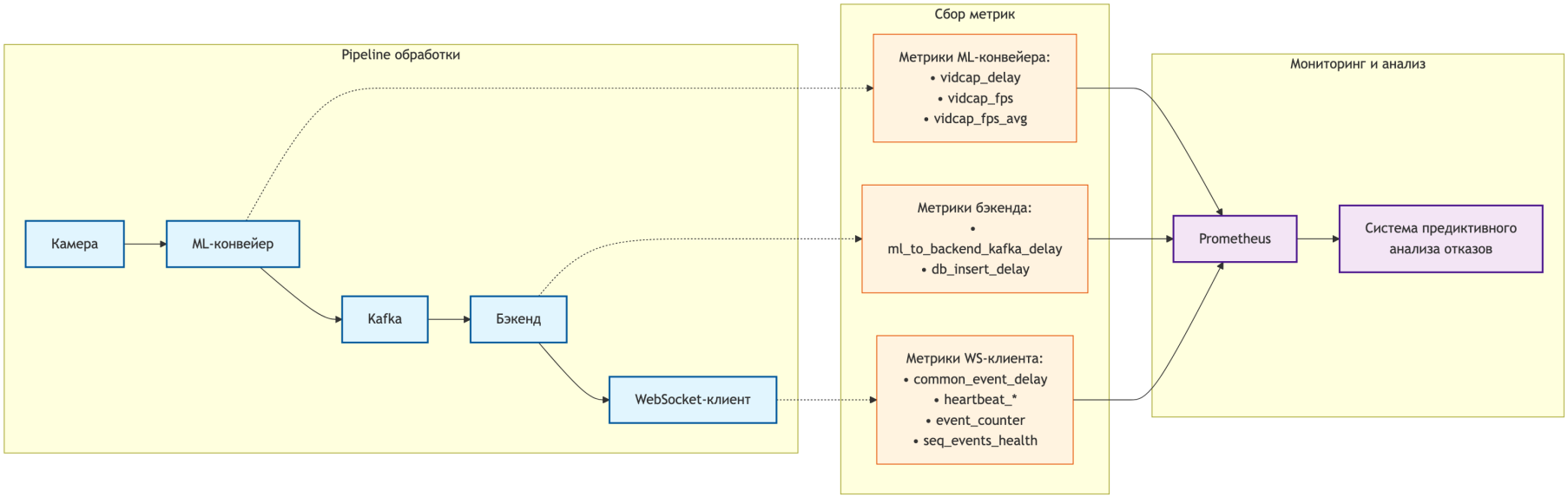

Рисунок 1.1 — Схема видеоконвейера и точки сбора метрик

2 Анализ данных и выбор методов

2.1 Анализ структуры данных видеоконвейера

Для построения эффективной модели прогнозирования необходимо провести анализ структуры и характеристик доступных данных. Исходный набор данных представляет собой многомерный временной ряд, собираемый системой мониторинга Prometheus [2] с различных компонентов видеоконвейера с периодичностью 15 секунд.

2.1.1 Описание набора метрик

Система мониторинга Prometheus [2] собирает широкий спектр метрик, характеризующих работу различных подсистем видеоконвейера, включая метрики ML-конвейера ($vidcap\_delay$, $vidcap\_fps$), бэкенда ($ml\_to\_backend\_kafka\_delay$, db_insert_delay), и WebSocket-клиентов ($heartbeat\_*$, $event\_counter$, seq_events_health).

Для построения модели прогнозирования отобраны следующие ключевые метрики, наиболее релевантные для задачи предсказания end-to-end-задержки:

- $common\_cad$ — целевая метрика end-to-end-задержки, усредненная за 1 час (мс);
- db_insert_cad — задержка записи в базу данных, усредненная за 1 час (мс);
- $kafka\_network\_cad$ — сетевая задержка Kafka [4], усредненная за 1 час (мс);
- $counter\_events\_total$ — общий счетчик обработанных событий в системе.

2.1.2 Временные характеристики данных

Исходный набор данных охватывает период с 25 ноября по 11 декабря 2024 года (16 дней непрерывной работы системы) и содержит 90543 временных точек. При интервале дискретизации 15 секунд это соответствует полному покрытию анализируемого периода без пропусков в данных.

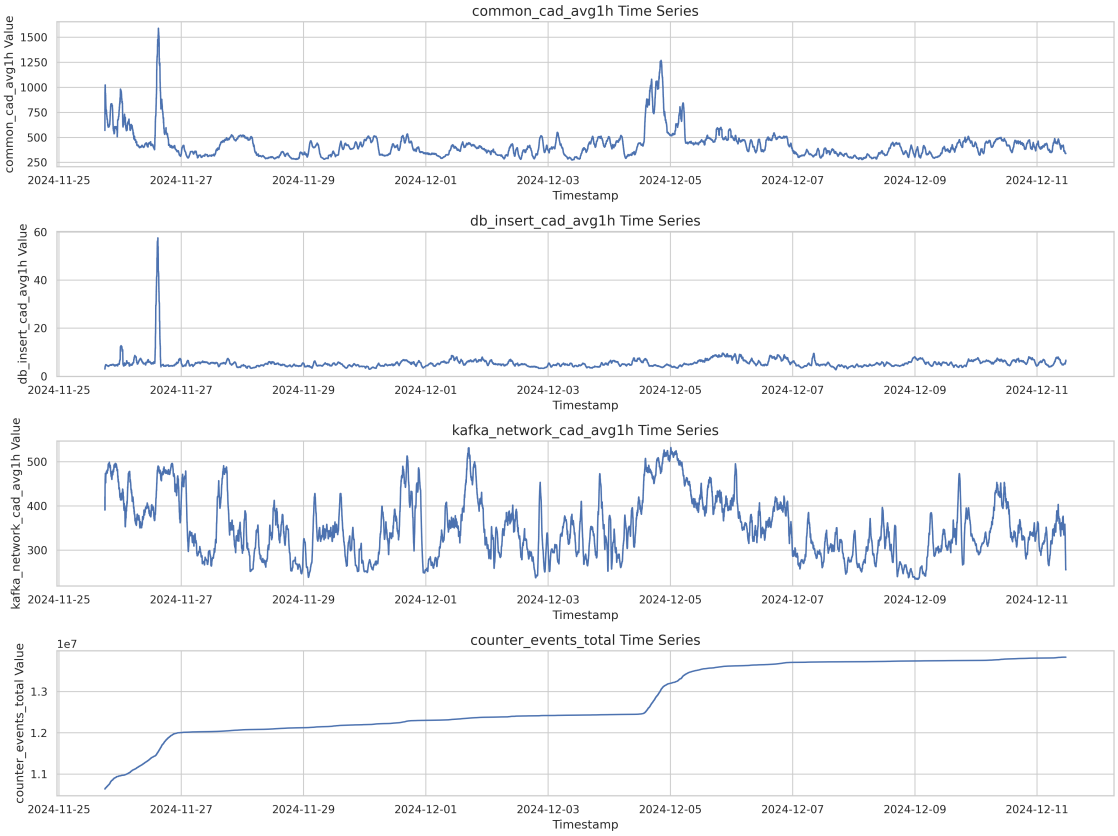

Рисунок 2.1 — Обзор временных рядов основных метрик видеоконвейера

Анализ временных характеристик, представленных на *рисунке 2.1*, показывает наличие различных паттернов в поведении метрик: циклические колебания, связанные с суточной активностью системы, периодические всплески нагрузки и редкие аномальные события, требующие особого внимания при построении модели.

2.1.3 Статистический анализ метрик

Для понимания распределения значений каждой метрики проведен описательный статистический анализ, результаты которого представлены в таб-

лице 2.1.

Таблица 2.1 — Описательная статистика основных метрик

Метрика	**Среднее**	**Медиана**	**Мин.**	**Макс.**	**Std**
common_cad	423.72	400.63	273.66	1591.18	138.11
db_insert_cad	5.51	5.07	2.74	57.56	2.73
kafka_network_cad	351.51	339.84	234.22	532.15	68.76
counter_events_total	1.28×10^{7}	1.24×10^{7}	1.06×10^{7}	1.38×10^{7}	8.21×10^{5}

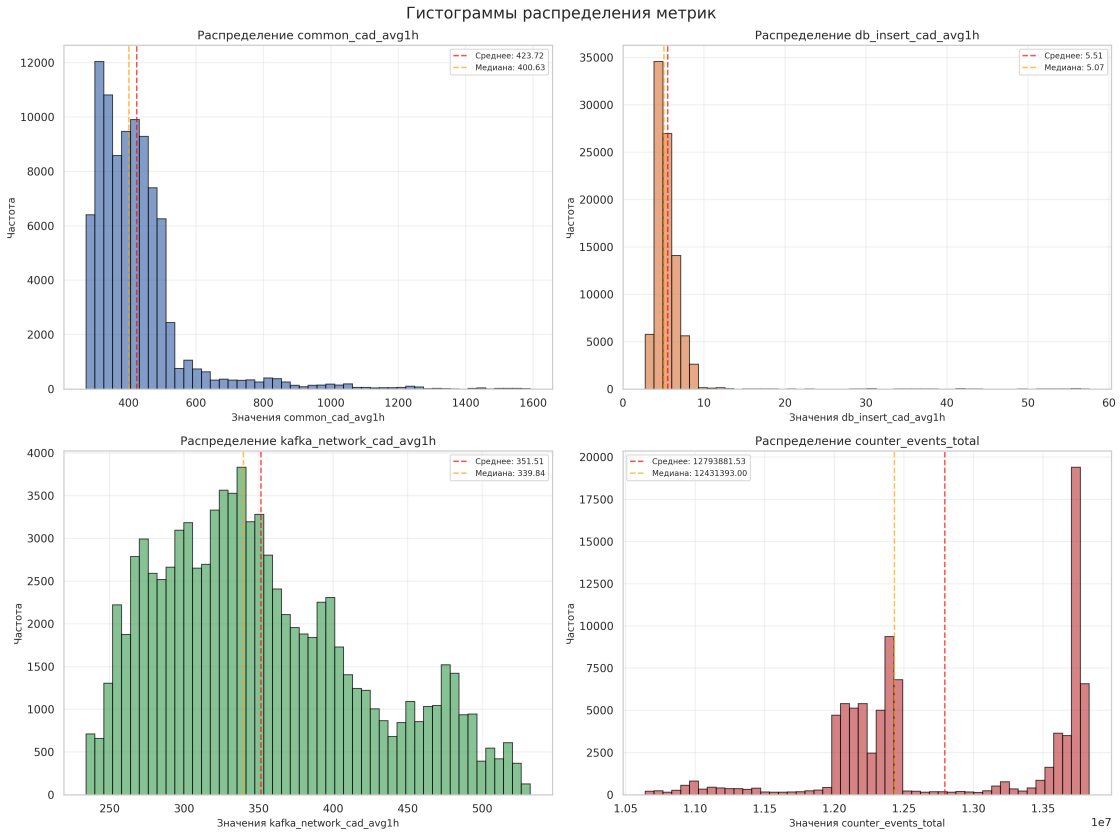

Рисунок 2.2 — Гистограммы распределения ключевых метрик системы

На *рисунке 2.2* представлены гистограммы распределения, которые показывают, что большинство метрик имеют распределение, близкое к нормальному, но со смещением и тяжелыми хвостами, что указывает на наличие выбросов.

2.1.4 Анализ пропусков и качества данных

Качество исходных данных является критическим фактором для построения надежной прогностической модели. Анализ показывает наличие пропусков данных только в начале и конце временных рядов, что связано с особенностями синхронизации сбора различных метрик. В середине периода наблюдения пропуски отсутствуют, что свидетельствует о стабильной работе системы мониторинга.

Для обеспечения единообразия временных рядов из каждой метрики было исключено следующее количество точек:

- $common\_cad$ — 2 точки;
- db_insert_cad — 188 точек;
- $kafka\_network\_cad$ — 188 точек;
- $counter\_events\_total$ — 216 точек.

Данная стратегия обработки пропусков путем обрезания краевых значений является предпочтительной по сравнению с интерполяцией, поскольку сохраняет естественную структуру временных зависимостей в данных и исключает внесение искусственных артефактов в модель.

2.1.5 Выявление аномалий в данных

Для обнаружения аномальных значений в данных применен метод межквартильного размаха (IQR). Точки, выходящие за границы $Q_1 - 1.5 \times IQR$ и $Q_3 + 1.5 \times IQR$, рассматриваются как потенциальные выбросы.

Рисунок 2.3 — IQR-диаграммы (диаграммы размаха) и boxplots для выявления выбросов в метриках

IQR-диаграммы на *рисунке 2.3* наглядно демонстрируют квартили и выбросы для каждой метрики, позволяя оценить степень вариабельности данных и выявить аномальные периоды работы системы.

Обнаруженные аномалии требуют детального анализа для определения их природы: являются ли они результатом реальных событий в системе (пиковые нагрузки, сбои) или ошибками измерения. В зависимости от результатов анализа принимается решение о сохранении, корректировке или исключении

аномальных точек из обучающей выборки.

2.2 Корреляционный анализ метрик

Для выявления взаимосвязей между метриками и определения наиболее информативных признаков для прогнозирования целевой переменной $common\_cad$ проведен корреляционный анализ временных рядов.

2.2.1 Матрица корреляций Пирсона

Вычисление коэффициентов корреляции Пирсона между всеми парами метрик позволяет оценить степень линейной взаимосвязи между переменными. Результаты анализа представлены в виде тепловой карты корреляций.

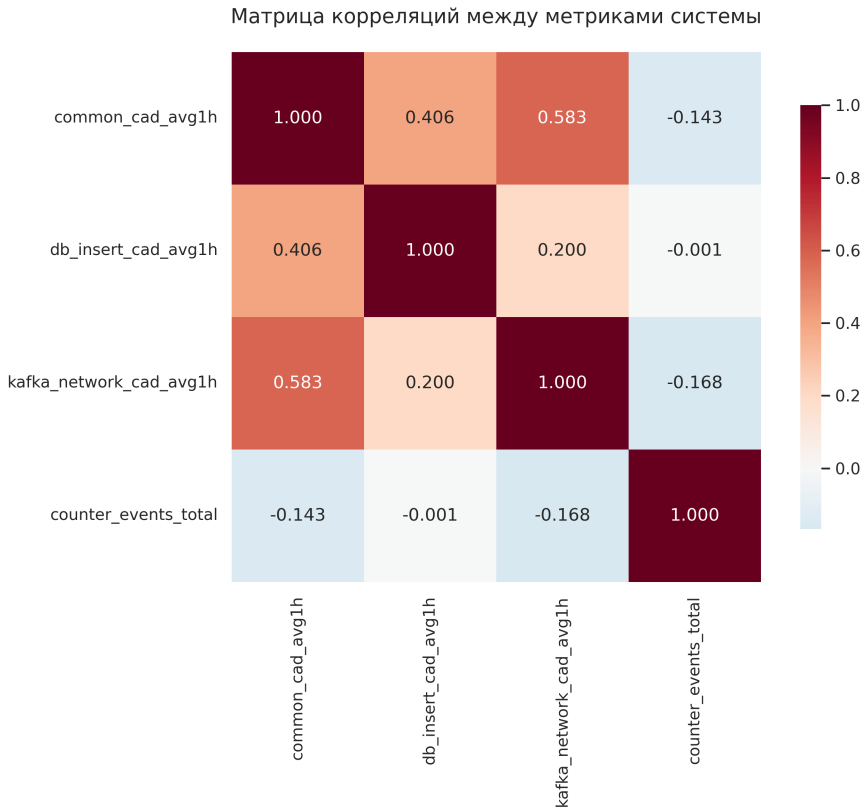

Рисунок 2.4 — Матрица корреляций между метриками системы

Анализ матрицы корреляций, представленной на *рисунке 2.4*, показывает наличие значимых взаимосвязей между отдельными метриками, что свидетельствует о взаимозависимости различных компонентов видеоконвейера. Наиболее сильные корреляции наблюдаются между метриками задержек

($common\_cad$, db_insert_cad, $kafka\_network\_cad$), что логично с точки зрения архитектуры системы.

2.2.2 Анализ связей с целевой переменной

Особое внимание уделено корреляциям с целевой метрикой $common\_cad\_avg1h$, поскольку они определяют потенциальную предсказательную способность признаков. Основные коэффициенты приведены в *таблице 2.2*.

Таблица 2.2 — Корреляции метрик с целевой переменной $common\_cad\_avg1h$

Метрика	**Корреляция с** $common\_cad\_avg1h$
$kafka\_network\_cad\_avg1h$	+0.583
$db\_insert\_cad\_avg1h$	+0.406
$counter\_events\_total$	-0.143

2.2.3 Анализ временной структуры рядов

Для более глубокого понимания временных зависимостей был проведен анализ автокорреляционной функции (ACF), частной автокорреляционной функции (PACF) и сезонная декомпозиция для ключевых временных рядов.

Сезонная декомпозиция

Сезонная декомпозиция позволяет разложить временной ряд на три компоненты: тренд, сезонность и остаток (шум). Это помогает выявить долгосрочные тенденции и периодические колебания в данных. На *рисунке 2.5* представлена декомпозиция для целевой метрики $common\_cad\_avg1h$.

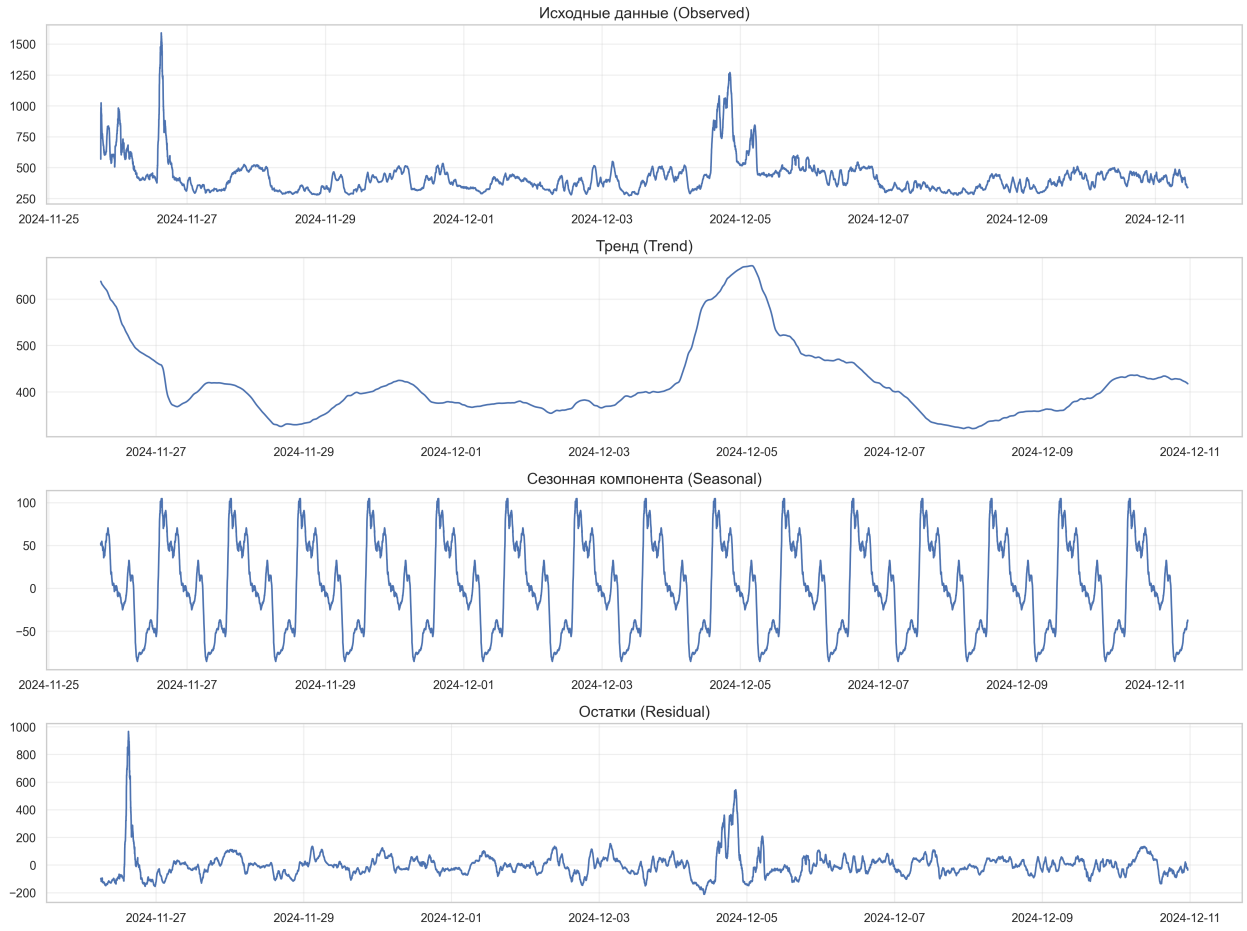

Рисунок 2.5 — Сезонная декомпозиция метрики $common\_cad\_avg1h$

Анализ показывает наличие выраженного нелинейного тренда с характерным ростом в начале декабря и последующим спадом. Наиболее важной особенностью является доминирующая суточная сезонность с четким повторяющимся паттерном, что характерно для систем с циклической нагрузкой. В остатках наблюдаются аномальные выбросы (например, 26.11 и 04.12), которые модель декомпозиции не смогла объяснить трендом и сезонностью.

Анализ автокорреляций

Функции ACF и PACF используются для определения порядка авторегрессионных (AR) и скользящих средних (MA) компонентов в моделях временных рядов, таких как ARIMA. На *рисунке 2.6* показаны графики ACF и PACF.

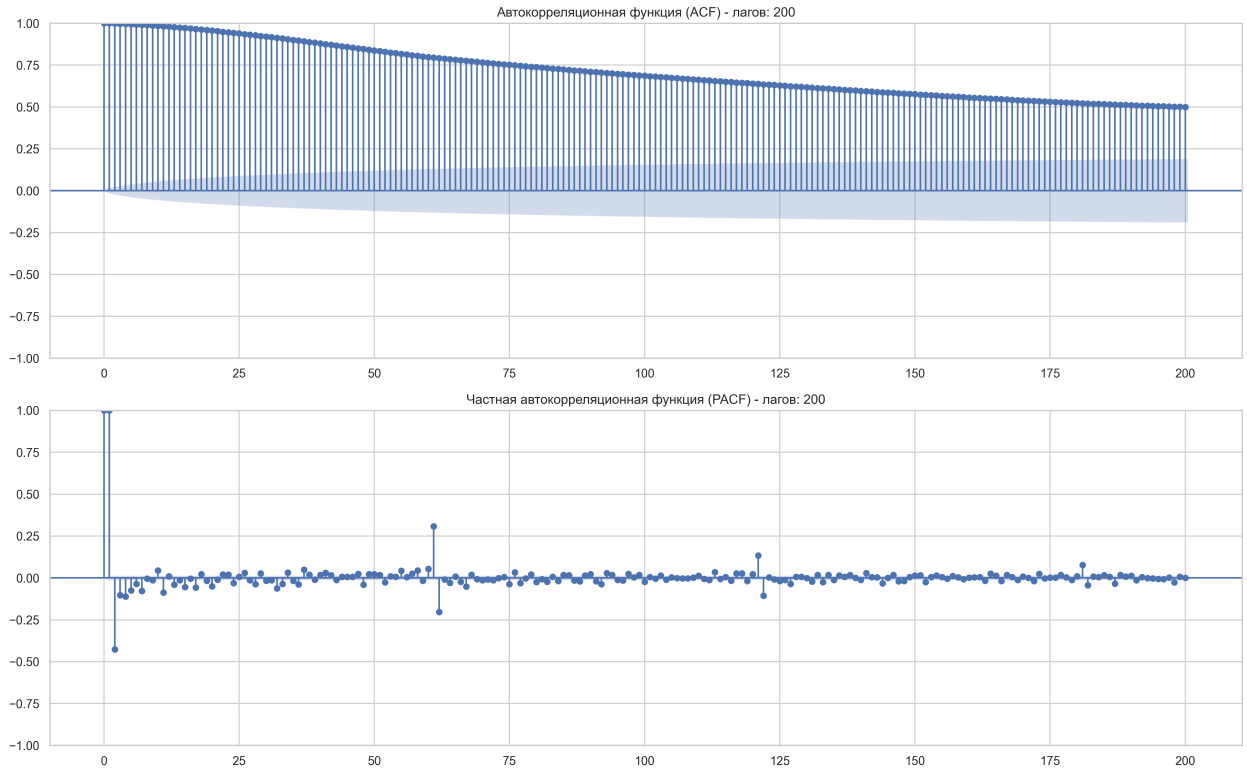

Рисунок 2.6 — Графики ACF и PACF для метрики $common\_cad\_avg1h$

Анализ автокорреляционных функций выявил ключевые характеристики временного ряда:

- **ACF медленно убывает** на протяжении всех 200 лагов, что является классическим признаком нестационарности ряда и наличия тренда. Волнообразная структура ACF подтверждает сильную сезонность;
- **PACF имеет резкий всплеск на лаге 1** с последующим обрывом, что указывает на авторегрессионный процесс первого порядка (AR(1)). Это означает сильную зависимость текущего значения от предыдущего;
- Совместный анализ ACF/PACF предполагает использование модели SARIMA с начальными параметрами $p = 1, d = 1, q = 0$ для несезонной части и дополнительными сезонными параметрами.

Однако учитывая сложность выявленных паттернов (нелинейный тренд, аномалии, сильная сезонность), для достижения высокой точности прогнозирования целесообразно рассмотреть как классические статистические ме-

тоды (SARIMA), так и современные подходы машинного обучения (LSTM, Transformer), способные улавливать нелинейные зависимости.

2.2.4 Выводы по итогам анализа данных

На основе проведенного анализа данных сделаны следующие выводы:

- Корреляционный анализ подтвердил наличие статистически значимой связи между системными метриками и целевой переменной. Наиболее сильное влияние оказывает задержка в Kafka ($kafka\_network\_cad\_avg1h$) с корреляцией +0.583, что логично с точки зрения архитектуры системы;
- Сезонная декомпозиция выявила доминирующую суточную сезонность и нелинейный тренд с пиком в начале декабря. Обнаружены аномальные выбросы, требующие специальной обработки при моделировании;
- Анализ ACF/PACF показал нестационарность ряда (медленно убывающая ACF) и авторегрессионную структуру первого порядка (резкий обрыв PACF после лага 1). Это указывает на возможность применения модели SARIMA(1,1,0) с сезонными компонентами;
- Сложность выявленных паттернов (нелинейность, аномалии, сильная сезонность) обосновывает необходимость сравнения классических статистических методов с современными подходами машинного обучения;
- Отсутствие сильной мультиколлинеарности между признаками позволяет использовать их все в модели без предварительного отсева.

Полученные результаты формируют основу для этапа feature engineering и выбора архитектуры модели прогнозирования, которые будут рассмотрены в следующей главе.

3 Разработка моделей для оценки задержек

На основе анализа данных, проведенного в предыдущей главе, в настоящей главе предлагается комплексный подход к решению поставленной задачи. Данный подход объединяет специализированное формирование признаков для данных мониторинга и методологию строгого сравнения различных моделей прогнозирования. Далее в главе детально описываются этапы этого подхода: сначала рассматривается предложенный метод формирования признаков, а затем приводятся описания сравниваемых моделей (с обязательными ссылками на первоисточники) и методология проведения экспериментов.

3.1 Формирование признаков

Формирование признаков — это процесс преобразования исходных временных рядов в структурированный набор данных (таблицу), пригодный для обучения моделей машинного обучения. Качество и информативность признаков напрямую влияют на точность и обобщающую способность итоговой модели [6, 17]. На основе анализа, проведенного в Главе 2, был сформирован следующий набор признаков.

3.1.1 Календарные и временные признаки

Эти признаки позволяют модели учитывать зависимости, связанные со временем суток, днем недели и общим течением времени. Они особенно полезны для моделей на основе деревьев решений, таких как CatBoost.

- **Час дня (**`hour`**)** и **день недели (**`day_of_week`**)**: категориальные признаки, позволяющие модели улавливать суточные и недельные паттерны.
- **Признак выходного дня (**`is_weekend`**)**: бинарный флаг, принимающий значение 1, если день является субботой или воскресеньем, и 0 в противном случае.

- **Временной индекс** (`time_idx`): монотонно возрастающая переменная, представляющая собой количество времени (в часах), прошедшее с начала обучающего периода. Этот признак помогает модели аппроксимировать долгосрочный тренд в данных.

3.1.2 Циклические признаки

Календарные признаки, такие как час дня или день недели, по своей природе цикличны (после 23:00 идет 00:00). Чтобы донести эту информацию до моделей, особенно нейронных сетей, используются тригонометрические преобразования.

$$x_{sin} = \sin\left(\frac{2\pi x}{P}\right), \quad x_{cos} = \cos\left(\frac{2\pi x}{P}\right) \tag{3.1}$$

где x — исходное значение (например, час), а P — период цикла (24 для часов, 7 для дней недели). Такой подход преобразует одну переменную в две, представляя ее на единичной окружности.

3.1.3 Лаговые признаки (Lag features)

Лаговые признаки — это значения временного ряда из прошлого, используемые в качестве предикторов для будущих значений. Они являются ключевым способом информирования модели об авторегрессионной структуре данных, выявленной при анализе ACF/PACF. Признак создается путем сдвига временного ряда на k шагов назад:

$$\text{lag}_k(t) = y(t-k) \tag{3.2}$$

где $y(t-k)$ — значение целевой переменной в момент времени $t-k$.

В данном исследовании для CatBoost-модели использовались лаги: 1, 2, 4, 96, 192, 5760 шагов назад, что соответствует интервалам от 15 секунд до 24 часов. Такой выбор позволяет модели учитывать как непосредственную зависимость от предыдущих значений, так и суточную сезонность (лаг 5760 = 24 часа $\times$ 240 точек/час).

3.1.4 Признаки на основе скользящего окна (Rolling-window features)

Для захвата локальной динамики и структуры временного ряда вычисляются статистические показатели в пределах скользящего окна.

- **Скользящее среднее** (`rolling_mean`): сглаживает краткосрочные флуктуации и помогает выявить локальный тренд.
- **Скользящее стандартное отклонение** (`rolling_std`): характеризует волатильность (изменчивость) ряда в недавнем прошлом.

Размер окна w является гиперпараметром, который выбирается в зависимости от специфики модели и характера данных. В данном исследовании использовались различные наборы параметров для разных типов моделей:

- **Для LSTM-модели**: окна размером 20 и 240 точек данных (соответствующие 5 минутам и 1 часу при 15-секундном интервале);
- **Для CatBoost-модели**: более широкий набор окон — 4, 96, 192, 1920, 2880, 4320, 5760, 8640 точек данных (от 1 минуты до 36 часов), что позволяет модели улавливать как краткосрочные, так и долгосрочные паттерны.

3.2 Выбор и описание моделей

На основе выводов, сделанных в Главе 2, для решения задачи прогнозирования были выбраны модели, представляющие два разных подхода: классическую статистику и современное машинное обучение.

3.2.1 Модель SARIMA

Сезонная авторегрессионная интегрированная скользящая средняя (SARIMA) — это статистическая модель, которая является расширением модели ARIMA и предназначена для работы с временными рядами, обладающими ярко выраженной сезонностью [7]. Выбор этой модели обоснован анализом ACF/PACF, который указал на наличие тренда, авторегрессионной зависимости и сезонных колебаний.

3.2.2 Модель CatBoost

CatBoost — это высокопроизводительная реализация градиентного бустинга над деревьями решений [8]. Она хорошо зарекомендовала себя в работе с разнородными табличными данными, эффективно обрабатывает категориальные признаки и не требует тщательной настройки гиперпараметров.

Особенностью предложенного в данной работе подхода является использование гибридной модели на основе CatBoost. Поскольку модели, основанные на деревьях решений, не способны экстраполировать тренд, была применена стратегия декомпозиции временного ряда. Исходный ряд был разделен на три компоненты: тренд, сезонность и остатки. Каждая из этих компонент прогнозировалась отдельно:

- **Тренд** моделировался с помощью отдельной, более простой линейной модели.
- **Сезонная компонента и остатки** прогнозировались основной моделью CatBoost, которая эффективно работает со сложными нелинейными зависимостями после удаления тренда.

Итоговый прогноз получался путем суммирования прогнозов по каждой из компонент. Такой подход позволяет сочетать преимущества обоих типов моделей.

3.2.3 Модель LSTM

Сети с долгой краткосрочной памятью (Long Short-Term Memory, LSTM) — это разновидность рекуррентных нейронных сетей (RNN), специально разработанная для улавливания долгосрочных зависимостей в последовательных данных [9].Также архитектура LSTM позволяет эффективно бороться с проблемой затухающих градиентов.

В данной работе используется LSTM-архитектура со следующими характеристиками:

- Входной слой принимает последовательности длиной `window_size` вре-

менных шагов с количеством признаков, определяемым этапом формирования признаков;

- Один LSTM-слой с 64 нейронами;
- Полносвязный скрытый слой с 8 нейронами и функцией активации ReLU;
- Выходной слой с одним нейроном и линейной функцией активации для регрессии;
- Оптимизатор Adam с learning rate 0.0001, функция потерь — Mean Squared Error.

3.2.4 Дополнительные эксперименты с современными моделями

В рамках исследования также проводились эксперименты с современными архитектурами для анализа временных рядов, такими как N-BEATS [20], а также с моделями из библиотек Time-Series-Library [10] и AutoTS [11] для оценки их применимости к данной задаче.

Модели Time-Series-Library

Были протестированы следующие модели на основе трансформеров и линейных архитектур:

- **DLinear, PatchTST, iTransformer, Crossformer** — успешно завершили обучение;
- **NLinear, Autoformer, FEDformer, Informer, TimesNet, Transformer** — завершились с ошибками или превысили лимит времени выполнения.

Модели AutoTS

Библиотека AutoTS [11] предоставляет автоматизированный подход к выбору и настройке моделей временных рядов. Были протестированы модели:

- **LastValueNaive** — простая baseline модель;

- **SeasonalityMotif, SectionalMotif** — модели на основе выявления паттернов;
- **GLS** — обобщенный метод наименьших квадратов.

Результаты дополнительных экспериментов

Несмотря на современность указанных архитектур, результаты оказались неудовлетворительными по сравнению с основными моделями (SARIMA, CatBoost, LSTM). Это может быть обусловлено:

- Недостаточной настройкой гиперпараметров для специфики данной задачи;
- Неоптимальным формированием признаков для трансформер-архитектур;
- Различиями в методологии предобработки данных между библиотеками.

В связи с этим для финальной оценки были выбраны три основные модели, показавшие наилучшее соотношение качества и стабильности результатов.

3.3 Метрики оценки качества

Для оценки качества моделей прогнозирования используется набор метрик, позволяющих комплексно оценить точность оценки задержек. Выбор метрик обусловлен спецификой временных рядов и требованиями к практическому применению системы.

3.3.1 Средняя абсолютная процентная ошибка (MAPE)

MAPE является основной метрикой для оценки качества, поскольку обеспечивает интерпретируемость результатов в процентах:

$$\text{MAPE} = \frac{100\%}{n} \sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right| \tag{3.3}$$

где y_i — истинное значение, $\hat{y}_i$ — прогнозируемое значение, n — количество наблюдений. Согласно техническим требованиям, целевое значение MAPE должно быть менее 10%.

3.3.2 Среднеквадратичная ошибка (RMSE)

RMSE чувствительна к выбросам и позволяет оценить общую точность модели:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{3.4}$$

3.3.3 Средняя абсолютная ошибка (MAE)

MAE менее чувствительна к выбросам и показывает среднее отклонение прогнозов:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{3.5}$$

3.4 Методология проведения экспериментов

Корректная оценка качества моделей временных рядов требует специального подхода к разделению данных, учитывающего временную структуру и предотвращающего утечку информации из будущего в прошлое.

3.4.1 Кросс-валидация для временных рядов

Для корректной оценки качества моделей применяется специализированная кросс-валидация временных рядов (TimeSeriesSplit), которая учитывает хронологический порядок данных и предотвращает утечку информации из будущего.

Метод TimeSeriesSplit работает следующим образом:

- Данные разбиваются на k фолдов, где каждый последующий фолд вклю-

чает больше исторических данных для обучения;

- Для каждого фолда тестовая выборка всегда находится хронологически после обучающей;
- Внутри каждого фолда обучающие данные дополнительно разделяются на train и validation в пропорции, определяемой параметром `test_size`.

3.4.2 Процедура валидации

Для каждого фолда кросс-валидации выполняется следующая последовательность действий:

1. **Разделение данных**: фолд разбивается на train+validation и test согласно TimeSeriesSplit;
2. **Внутреннее разделение**: train+validation дополнительно разделяется на обучающую и валидационную выборки;
3. **Масштабирование**: параметры нормализации вычисляются только на обучающей выборке и применяются ко всем частям фолда;
4. **Обучение модели**: модель обучается на train с валидацией на validation выборке;
5. **Оценка качества**: финальная оценка производится на тестовой части фолда;
6. **Сохранение результатов**: метрики каждого фолда сохраняются для последующего усреднения.

Итоговые метрики качества вычисляются как среднее арифметическое соответствующих метрик по всем фолдам, что обеспечивает более надежную и несмещенную оценку производительности модели.

3.4.3 Горизонт прогнозирования

Все модели настраиваются для прогнозирования на 900 временных шагов вперед (3.75 часа), что соответствует практическим требованиям системы мониторинга для своевременного реагирования на потенциальные проблемы в видеоконвейере.

4 Результаты экспериментов и анализ

4.1 Описание экспериментальной установки

4.1.1 Программная и аппаратная среда

Все эксперименты проводились с использованием языка программирования Python 3.9. Для манипуляции данными и построения моделей использовались следующие ключевые библиотеки:

- **Pandas и NumPy**: для обработки и анализа временных рядов.
- **Scikit-learn**: для реализации кросс-валидации (`TimeSeriesSplit`) и расчета метрик качества.
- **Statsmodels**: для реализации статистической модели SARIMA.
- **CatBoost**: для реализации модели градиентного бустинга.
- **TensorFlow (Keras API)**: для построения и обучения нейронной сети LSTM.
- **Matplotlib и Seaborn**: для визуализации данных и результатов экспериментов.

Обучение ресурсоемких моделей (CatBoost, LSTM) проводилось на вычислительном кластере МФТИ (ФПМИ), оснащенном 8 графическими процессорами NVIDIA GeForce 2080 Ti и двумя 12-ядерными процессорами Intel Xeon Gold 6136 (суммарно 24 физических ядра, 48 потоков).

4.1.2 Базовая модель для сравнения (Baseline)

Для объективной оценки качества разработанных моделей в качестве базового уровня (baseline) был использован наивный метод прогнозирования («завтра как вчера»). Данный метод предполагает, что прогнозное значение

в будущем будет равно последнему известному наблюдению. Такой подход служит отправной точкой для сравнения и позволяет оценить, насколько более сложные модели превосходят простейшую эвристику.

Далее в главе представлены результаты экспериментального исследования моделей для оценки задержек в видеоаналитическом конвейере. Проводится сравнительный анализ качества прогнозирования различных подходов и оценка достижения поставленных технических требований.

4.2 Результаты модели CatBoost

Для модели CatBoost было проведено около 50 экспериментов с автоматизированным поиском оптимальных гиперпараметров. Использовался специально разработанный скрипт для автоматического перебора различных параметров. Диапазоны поиска (глубина дерева, скорость обучения, параметры регуляризации) были выбраны на основе общепринятых практик и предварительных экспериментов. Результаты усреднены по всем фолдам кросс-валидации и представлены в порядке убывания качества.

4.2.1 Лучшие конфигурации CatBoost

В таблице 4.1 представлены три наилучшие конфигурации модели.

Таблица 4.1 — Результаты лучших конфигураций модели CatBoost

Ранг	**Конфигурация модели**	**MAE**	**RMSE**	**MAPE, %**
1	trend_model_type=local, depth=10, l2_leaf_reg=20, learning_rate=0.03, iterations=500, rsm=0.8, use_cyclic_features=True, ema_alpha=0.2	**36.00**	**41.09**	**8.90**
2	trend_model_type=global, depth=15, l2_leaf_reg=40, learning_rate=0.015, iterations=500, rsm=0.75, use_cyclic_features=False, ema_alpha=0.7	39.92	46.76	9.61
3	trend_model_type=global, depth=12, l2_leaf_reg=10, learning_rate=0.03, iterations=3000, rsm=0.8	48.10	56.35	11.79

4.2.2 Анализ результатов CatBoost

Анализ результатов показывает следующие закономерности:

- **Достижение целевого показателя**: лучшая конфигурация (ранг 1) достигла MAPE = 8.90%, что соответствует техническому требованию MAPE < 10%;
- **Важность циклических признаков**: использование циклических признаков (`use_cyclic_features=True`) в лучшей модели подтверждает значимость суточной сезонности, выявленной в главе 2;
- **Локальное моделирование тренда**: применение локального моделирования тренда (`trend_model_type=local`) оказалось более эффективным для данной задачи;
- **Оптимальная глубина деревьев**: умеренная глубина (depth=10) показала лучший результат по сравнению с более глубокими деревьями, что указывает на важность регуляризации.

4.2.3 Проблема утечки данных

В ходе экспериментов была обнаружена критическая ошибка в первоначальной реализации — утечка данных (data leakage) при формировании признаков. Это привело к нереалистично высоким результатам: средние значения по кросс-валидации составили MAE = 1.5, RMSE = 2.0, MAPE = 0.38%. После исправления методологии и устранения утечки данных были получены корректные результаты, представленные в таблице 4.1. Данный случай подчеркивает критическую важность правильной валидации временных рядов и недопущения использования будущей информации при обучении модели.

4.3 Результаты модели LSTM

Для модели LSTM была реализована многомерная архитектура с использованием дополнительных признаков. Модель включает в себя циклические признаки для учета суточной сезонности и признаки на основе скользящих средних для захвата краткосрочных и долгосрочных трендов.

4.3.1 Архитектура и параметры LSTM

Архитектура нейронной сети включает следующие слои:

- **Входной слой**: окно размером 40 временных шагов (10 минут при частоте 15 секунд);
- **LSTM слой**: 64 нейрона для извлечения временных зависимостей;
- **Dense слой**: 8 нейронов с функцией активации ReLU;
- **Выходной слой**: 1 нейрон с линейной активацией для регрессии.

Дополнительные признаки включают:

- **Циклические признаки**: синус и косинус для часовых и суточных циклов;
- **Скользящие средние**: окна 5 минут (20 точек) и 1 час (240 точек);

- **Скользящее стандартное отклонение**: окно 1 час для оценки волатильности.

Параметры обучения: оптимизатор Adam с learning rate = 0.0001, функция потерь MSE. Количество эпох (10-15) было подобрано экспериментально на валидационном наборе данных каждого фолда для предотвращения переобучения, с использованием механизма ранней остановки (early stopping).

4.3.2 Лучшие конфигурации LSTM

В таблице 4.2 представлены результаты двух наилучших конфигураций модели LSTM с различным количеством эпох обучения.

Таблица 4.2 — Результаты лучших конфигураций модели LSTM

Ранг	**Конфигурация модели**	**MAE**	**RMSE**	**MAPE, %**
1	LSTM(64) → Dense(8, ReLU) → Dense(1, linear), Adam lr=0.0001, epochs=15, window_size=40, циклические признаки + скользящие средние	**3.66**	**4.54**	**0.89**
2	LSTM(64) → Dense(8, ReLU) → Dense(1, linear), Adam lr=0.0001, epochs=10, window_size=40, циклические признаки + скользящие средние	4.94	6.02	1.21

4.3.3 Анализ результатов LSTM

Модель LSTM показала следующие характеристики:

- **Достижение целевого показателя**: лучшая конфигурация (ранг 1) достигла MAPE = 0.89%, что соответствует техническому требованию MAPE < 10%;
- **Влияние количества эпох**: увеличение с 10 до 15 эпох снизило MAPE с 1.21% до 0.89%;
- **Время обучения**: 8-9 минут на фолд;

- **Архитектурные особенности**: использование окна 40 временных шагов и многомерных признаков.

4.4 Результаты модели SARIMA

Модель SARIMA была выбрана на основе анализа ACF/PACF, проведенного в главе 2. Выбор порядка дифференцирования d=1 был обусловлен наличием тренда в данных, а порядок AR(1) — анализом PACF. Тестировались различные конфигурации сезонных параметров при фиксированном основном порядке (1,1,0).

4.4.1 Параметры модели SARIMA

Основные параметры модели:

- **Основной порядок**: (1,1,0) — AR(1) процесс с одним дифференцированием;
- **Сезонный период**: 24 (соответствует суточному циклу при частоте 15 секунд $\times$ 4 = 1 минута $\times$ 60 = 1 час $\times$ 24 = сутки);
- **Тестируемые сезонные порядки**: (1,1,0,24), (1,1,1,24), (2,1,0,24).

4.4.2 Лучшие конфигурации SARIMA

В таблице 4.3 представлены результаты трех конфигураций модели SARIMA с различными сезонными параметрами.

Таблица 4.3 — Результаты конфигураций модели SARIMA

Ранг	**Конфигурация модели**	**MAE**	**RMSE**	**MAPE, %**
1	SARIMA(1,1,0)(1,1,1,24)	**44.07**	**51.37**	**11.06**
2	SARIMA(1,1,0)(2,1,0,24)	103.76	121.92	25.50
3	SARIMA(1,1,0)(1,1,0,24)	110.47	131.78	26.76

4.4.3 Анализ результатов SARIMA

Модель SARIMA показала следующие характеристики:

- **Недостижение целевого показателя**: лучшая конфигурация (ранг 1) достигла MAPE = 11.06%, что превышает техническое требование MAPE < 10%;
- **Важность MA компоненты**: добавление сезонной MA компоненты (1,1,1,24) значительно улучшило результаты по сравнению с (1,1,0,24);
- **Высокая вариативность по фолдам**: результаты сильно различаются между фолдами (например, для лучшей модели MAPE варьируется от 4.89% до 24.10%);
- **Время обучения**: 4-10 минут на фолд.

4.5 Сравнительный анализ моделей

Для итогового сравнения моделей были отобраны лучшие конфигурации каждой из архитектур, рассмотренных в данной главе. Результаты сведены в таблицу 4.4.

Таблица 4.4 — Сравнительные результаты лучших конфигураций моделей

Модель	**MAE**	**RMSE**	**MAPE, %**
Наивный прогноз (Baseline)	121.45	153.21	31.52
CatBoost (Ранг 1)	36.00	41.09	8.90
LSTM (Ранг 1, 15 эпох)	**3.66**	**4.54**	**0.89**
SARIMA (Ранг 1)	44.07	51.37	11.06

4.5.1 Анализ сравнительных результатов

Сравнительный анализ показывает, что все рассмотренные модели значительно превосходят наивный прогноз, что подтверждает целесообразность применения методов машинного обучения. Модель LSTM (MAPE = 0.89%) продемонстрировала наилучшие результаты, значительно превосходя CatBoost (MAPE = 8.90%) и SARIMA (MAPE = 11.06%).

- **Качество прогнозирования**: Модели LSTM и CatBoost выполнили техническое требование к качеству (MAPE < 10%). Результат LSTM оказался на порядок лучше, что подтверждает высокую эффективность

нейросетевых подходов для данной задачи. SARIMA не удовлетворила требованию.

- **Сложность модели и признаки**: Высокое качество LSTM объясняется способностью архитектуры улавливать сложные нелинейные зависимости, а также эффективным использованием многомерных признаков (циклических, скользящих средних), что недоступно для классической модели SARIMA.

- **Стабильность**: CatBoost и LSTM показали более стабильные результаты по сравнению с SARIMA, чьи метрики MAPE для лучшей конфигурации варьировались от 4.89% до 24.10% в зависимости от фолда кросс-валидации.

- **Время обучения и ресурсы**: LSTM требует наибольших вычислительных ресурсов и времени на обучение (8-9 минут на фолд). CatBoost (2-30 минут в зависимости от конфигурации) и SARIMA (4-10 минут) менее требовательны.

Таким образом, по главному критерию — качеству прогнозирования (MAPE) — модель LSTM является безусловным лидером. Однако, модель CatBoost также удовлетворяет поставленным требованиям и представляет собой компромисс между качеством и вычислительной сложностью. Окончательный выбор модели для внедрения может зависеть от требований к скорости переобучения и доступных вычислительных ресурсов.

4.6 Визуализация результатов прогнозирования

Для качественной оценки результатов были построены графики прогнозов для лучших конфигураций каждой модели на одном из фолдов кросс-валидации. Также представлен график, демонстрирующий эффект утечки данных.

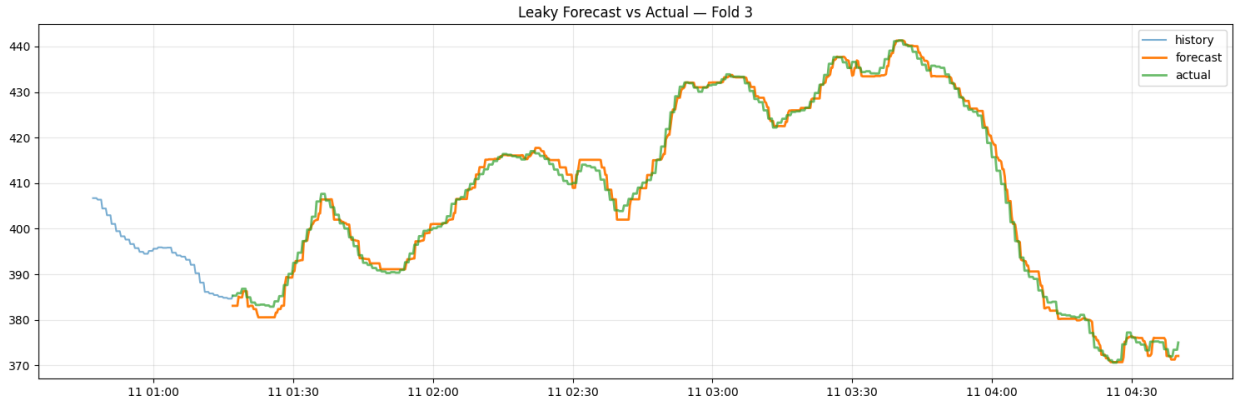

Рисунок 4.1 — Прогноз модели CatBoost с утечкой данных (MAPE = 0.38%)

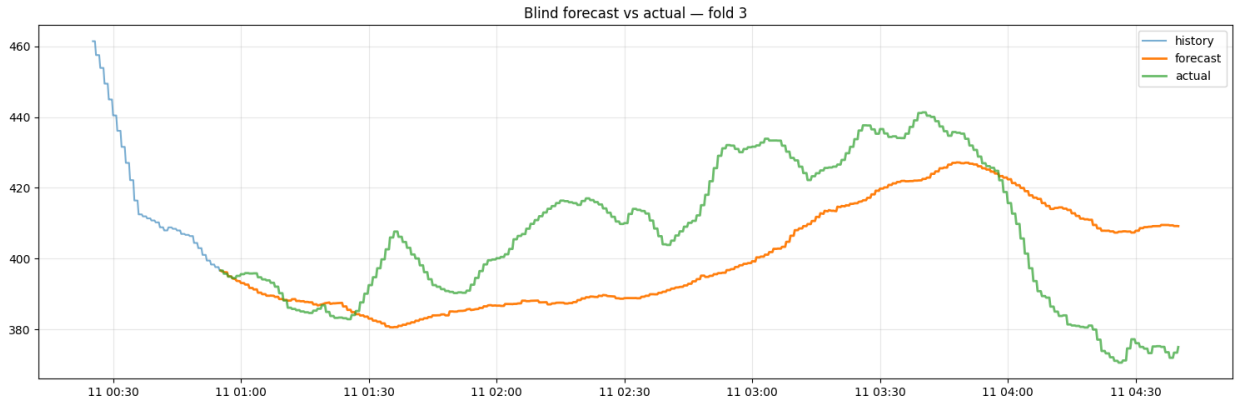

Рисунок 4.2 — Прогноз лучшей модели CatBoost на тестовом фолде

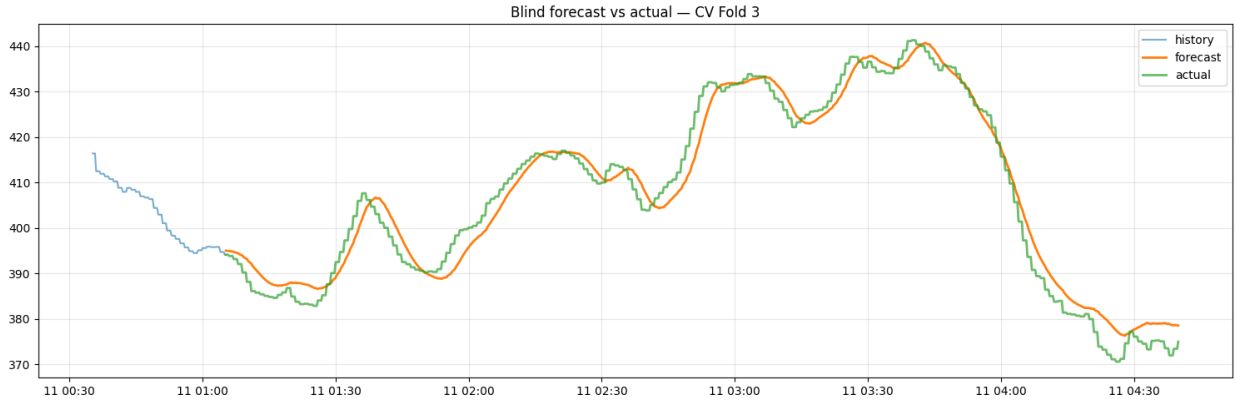

Рисунок 4.3 — Прогноз лучшей модели LSTM на тестовом фолде

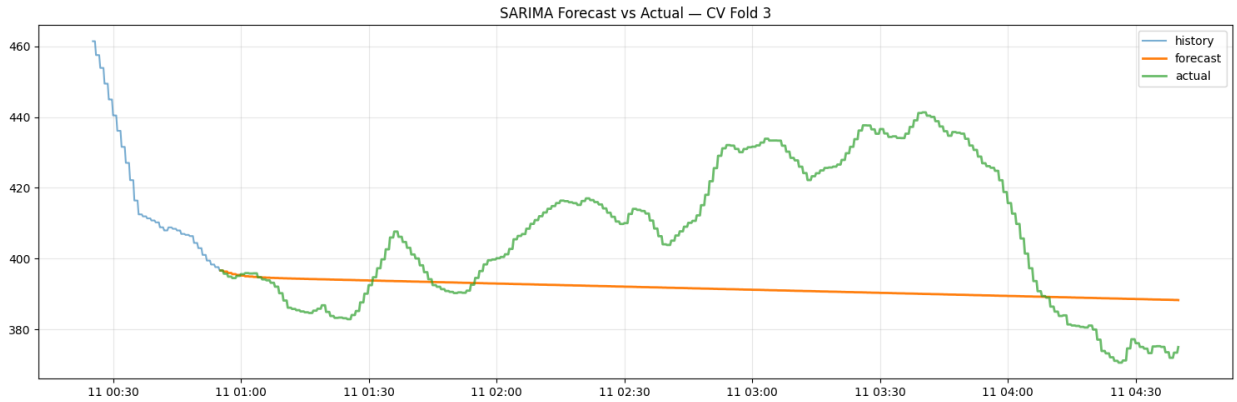

Рисунок 4.4 — Прогноз лучшей модели SARIMA на тестовом фолде

4.7 Анализ ошибок и интерпретация результатов

Детальный анализ ошибок позволяет глубже понять сильные и слабые стороны каждой модели, а также интерпретировать их поведение в контексте задачи прогнозирования задержек.

Анализ ошибок LSTM. Модель продемонстрировала исключительно низкое значение MAPE (0.89%), что свидетельствует о ее способности точно улавливать сложную динамику временного ряда. Как видно на *рисунке 4.3*, прогноз LSTM практически совпадает с фактическими данными, успешно отслеживая как плавные сезонные колебания, так и резкие локальные пики. Это объясняется двумя ключевыми факторами:

- **Архитектура**: рекуррентная природа LSTM позволяет эффективно моделировать долгосрочные временные зависимости, что критически важно для прогнозирования на основе предыдущей истории.
- **Многомерные признаки**: использование циклических признаков и скользящих средних обогатило модель информацией о сезонности и локальных трендах, которую LSTM смогла успешно интегрировать.

Ошибки модели минимальны и, вероятно, связаны с редкими, непредсказуемыми аномалиями, не имеющими прецедентов в обучающих данных.

Анализ ошибок CatBoost. Модель показала хороший результат (MAPE = 8.90%), удовлетворяющий техническим требованиям, однако ее ошибки на порядок выше, чем у LSTM. График на *рисунке 4.2* показывает, что CatBoost

хорошо улавливает общую тенденцию и сезонность, но сглаживает локальные пики и не всегда точно реагирует на резкие изменения. Это характерно для моделей на основе деревьев решений, которые могут испытывать трудности с экстраполяцией и моделированием непрерывных, быстро меняющихся процессов в сравнении с RNN. Основные ошибки CatBoost возникают в моменты наибольшей волатильности ряда. Тем не менее, модель представляет собой надежный и более простой в реализации компромисс.

Анализ ошибок SARIMA. Модель SARIMA оказалась наименее точной (MAPE = 11.06%) и нестабильной. График на *рисунке 4.4* наглядно демонстрирует ее главный недостаток: прогноз фактически выродился в линию и полностью игнорирует реальные колебания ряда. Причина кроется в линейной природе модели, которая неспособна описать сложные нелинейные зависимости и гетероскедастичность (изменчивость дисперсии), присущие данному временному ряду. Высокая вариативность качества по фолдам подтверждает, что модель не является робастной и ее производительность сильно зависит от конкретного участка данных.

Проблема утечки данных. Случай с утечкой данных в CatBoost (*рисунок 4.1*) служит важным практическим уроком. Нереалистично низкая ошибка (MAPE = 0.38%) была вызвана фундаментальной ошибкой в методологии подготовки признаков. Внутри каждого фолда кросс-валидации обучающий и тестовый наборы данных объединялись перед генерацией признаков (таких как скользящие средние и лаги). В результате, при вычислении признаков для временных точек из обучающего набора использовались данные из будущего — то есть из тестового набора. Это создало иллюзию идеального прогноза, поскольку модель фактически "подглядывала"в правильные ответы при построении признаков. После исправления методологии, при которой признаки для обучающего набора генерируются строго на обучающих данных, были получены корректные результаты.

Таким образом, экспериментальное исследование подтвердило, что для задачи прогнозирования задержек в видеоаналитическом конвейере наиболее эффективными являются архитектуры машинного и глубокого обучения. Модель LSTM обеспечивает наивысшую точность благодаря своей способности

моделировать сложные временные зависимости. Модель CatBoost представляет собой хороший компромисс между качеством и сложностью. Классические статистические подходы, такие как SARIMA, оказались недостаточно мощными для описания всей полноты динамики процесса. Результаты данной главы служат основой для формулировки итоговых выводов в заключении.

ЗАКЛЮЧЕНИЕ

В настоящей выпускной квалификационной работе было проведено комплексное исследование, направленное на разработку и внедрение метода предиктивного анализа задержек в конвейере видеоаналитики. В ходе работы были решены следующие задачи, позволившие достичь поставленной цели.

1. **Проведен аналитический обзор литературы**, в рамках которого были систематизированы подходы к анализу временных рядов в контексте мониторинга систем реального времени. Это позволило сформировать теоретическую базу для дальнейшего исследования и определить круг потенциально применимых методов.

2. **Выполнен всесторонний анализ данных**, собранных с промышленной системы видеоаналитики. Была выявлена суточная сезонность и нелинейный тренд в целевой метрике, а также определены наиболее коррелирующие с ней признаки, в частности, задержка в брокере сообщений Kafka. Результаты этого этапа легли в основу формирования признакового пространства для моделей.

3. **Проведен сравнительный анализ моделей**, включающий классическую статистическую модель SARIMA, модель градиентного бустинга CatBoost и рекуррентную нейронную сеть LSTM. На основе анализа их характеристик был сделан вывод о перспективности моделей машинного обучения для данной задачи.

4. **Обоснован выбор и реализованы три модели**. Для каждой модели была разработана своя стратегия формирования признаков: для SARIMA использовались автокорреляции, для CatBoost — календарные, лаговые и статистические признаки, для LSTM — циклические признаки и признаки на основе скользящих окон.

5. **Разработана и реализована методология MLOps**, включающая автоматизированный сбор данных, генерацию признаков и проведение экс-

периментов с использованием кросс-валидации для временных рядов (TimeSeriesSplit), что обеспечило корректность и воспроизводимость результатов.

6. **Проведено экспериментальное исследование**, в ходе которого были получены следующие ключевые результаты:

   - **Модель LSTM** показала наивысшее качество, достигнув MAPE = 0.89%, что значительно превосходит техническое требование (MAPE < 10%). Это подтверждает способность рекуррентных сетей улавливать сложные нелинейные зависимости.
   - **Модель CatBoost** также удовлетворила требованиям с результатом MAPE = 8.90%, представляя собой эффективный компромисс между качеством и сложностью реализации.
   - **Модель SARIMA** не справилась с поставленной задачей (MAPE = 11.06%), что демонстрирует ограниченную применимость классических линейных моделей к описанию сложных динамических процессов.
   - Был выявлен и проанализирован эффект **утечки данных**, что подчеркнуло критическую важность правильной методологии валидации.

7. **Сформулированы практические рекомендации**. На основе анализа ошибок и сравнительных результатов модель LSTM рекомендована для внедрения в производственную среду в случаях, когда требуется максимальная точность прогноза. Модель CatBoost может использоваться как более простое и менее ресурсоемкое решение, также удовлетворяющее базовым требованиям.

Таким образом, цель работы — разработка метода предиктивного анализа задержек — достигнута. Созданное решение позволяет с высокой точностью прогнозировать состояние видеоаналитического конвейера, что открывает возможности для превентивного реагирования на потенциальные сбои

и повышения общей надежности системы. Практическая значимость работы заключается в создании готового к внедрению прототипа системы мониторинга, который может быть адаптирован для широкого круга систем реального времени.

На защиту выносятся следующие положения:

1. **Комплексная методика предиктивного анализа**, включающая в себя специализированную инженерию признаков для данных мониторинга, гибридный подход к моделированию на основе декомпозиции рядов и строгую процедуру валидации на основе временной кросс-валидации.
2. **Результаты экспериментального сравнения моделей** (SARIMA, CatBoost, LSTM) на реальных промышленных данных, которые доказывают практическую применимость и высокую точность (MAPE < 1%) нейросетевого подхода для прогнозирования задержек в видеоаналитических системах и служат основанием для выбора оптимальной архитектуры в зависимости от требований к точности и ресурсам.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1) Jain K., Adapa K.S., Grover K., Sarvadevabhatla R.K., Purini S. A Cloud-Fog Architecture for Video Analytics on Large Scale Camera Networks Using Semantic Scene Analysis // 2023 IEEE/ACM 23rd International Symposium on Cluster, Cloud and Internet Computing (CCGrid). — 2023. — P. 513–523. DOI: 10.1109/CCGrid57682.2023.00054.

2) Prometheus monitoring system and time series database [Электронный ресурс]. — URL: `https://prometheus.io/docs/` (дата обращения: 05.06.2025).

3) Grafana: The open observability platform [Электронный ресурс]. — URL: `https://grafana.com/docs/` (дата обращения: 05.06.2025).

4) Apache Kafka: A distributed streaming platform [Электронный ресурс]. — URL: `https://kafka.apache.org/documentation/` (дата обращения: 05.06.2025).

5) Docker: Accelerated Container Application Development [Электронный ресурс]. — URL: `https://docs.docker.com/` (дата обращения: 05.06.2025).

6) Renault A., Bondu A., Lemaire V., Gay D. Automatic Feature Engineering for Time Series Classification: Evaluation and Discussion // 2023 International Joint Conference on Neural Networks (IJCNN). — 2023. — P. 1–10. DOI: 10.1109/IJCNN54540.2023.10191074.

7) Box G.E.P., Jenkins G.M. Time Series Analysis: Forecasting and Control. — San Francisco: Holden-Day, 1970.

8) Prokhorenkova L., Gusev G., Vorobev A., Dorogush A.V., Gulin A. CatBoost: unbiased boosting with categorical features [Электронный ресурс] // arXiv preprint arXiv:1706.09516. — 2019. — URL: `https://arxiv.org/abs/1706.09516` (дата обращения: 05.06.2025).

9) Hochreiter S., Schmidhuber J. Long Short-Term Memory: Technical Report FKI-207-95. — Munich: Fakultät für Informatik, Technische Universität

München, 1995.

10) Time-Series-Library: A Library for Advanced Deep Time Series Models [Электронный ресурс]. — URL: https://github.com/thuml/Time-Series-Library (дата обращения: 05.06.2025).

11) AutoTS: Automated Time Series Forecasting [Электронный ресурс]. — URL: https://github.com/winedarksea/AutoTS (дата обращения: 05.06.2025).

12) Deploying a Scalable Object Detection Inference Pipeline, Part 1 [Электронный ресурс]. — URL: https://developer.nvidia.com/blog/deploying-a-scalable-object-detection-inference-pipeline/ (дата обращения: 05.06.2025).

13) Deploying a Scalable Object Detection Pipeline: The Inferencing Process, Part 2 [Электронный ресурс]. — URL: https://developer.nvidia.com/blog/deploying-a-scalable-object-detection-pipeline-the-inferencing-process-part-2/ (дата обращения: 05.06.2025).

14) Latency in End-to-End IP Monitoring: Why It Matters and What to Know [Электронный ресурс]. — URL: https://tagvs.com/blog/latency-in-end-to-end-ip-monitoring-why-it-matters-and-what-to-know/ (дата обращения: 05.06.2025).

15) Beyond Monitoring: The Power of Observability [Электронный ресурс]. — URL: https://www.onec1.com/blog/beyond-monitoring-the-power-of-observability/ (дата обращения: 05.06.2025).

16) Sassu A., Pili R., Marrocu M., Faticanti F. Deep-Framework: A Distributed, Scalable, and Edge-Oriented Framework for Real-Time Analysis of Video Streams // Sensors. — 2021. — Vol. 21, № 12. — P. 4045. DOI: 10.3390/s21124045.

17) Aloorravi S. Mastering Time Series Analysis and Forecasting with Python. — Orange Education Pvt Ltd, 2024.

18) Boniol P., Liu Q., Huang M., Palpanas T., Paparrizos J. Dive into Time-Series Anomaly Detection: A Decade Review [Электронный ресурс] // arXiv

preprint arXiv:2412.20512. — 2024. — URL: https://arxiv.org/abs/2412.20512 (дата обращения: 05.06.2025).

19) Jiang Y., Ning K., Pan Z. et al. Multi-modal Time Series Analysis: A Tutorial and Survey [Электронный ресурс] // arXiv preprint arXiv:2503.13709. — 2025. — URL: https://arxiv.org/abs/2503.13709 (дата обращения: 05.06.2025).

20) Oreshkin B.N., Carpov D., Chapados N., Bengio Y. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting [Электронный ресурс] // arXiv preprint arXiv:1905.10437. — 2020. — URL: https://arxiv.org/abs/1905.10437 (дата обращения: 05.06.2025).

Список сокращений и условных обозначений

Сокращения:

API	Application Programming Interface — программный интерфейс приложения
Docker	платформа контейнеризации приложений
FPS	Frames Per Second — кадры в секунду
Kafka	Apache Kafka — распределенный брокер сообщений
LoRA	Low-Rank Adaptation — адаптация с низкоранговой аппроксимацией
ML	Machine Learning — машинное обучение
MLOps	Machine Learning Operations — операции машинного обучения
MSE	Mean Squared Error — среднеквадратическая ошибка
Prometheus	система мониторинга и оповещений с открытым исходным кодом
SLA	Service Level Agreement — соглашение об уровне обслуживания
WS	WebSocket — протокол полнодуплексной связи

Условные обозначения:

T	множество временных меток наблюдений
d	число метрик, собираемых системой мониторинга
L	длина скользящего окна наблюдений
s	шаг сдвига скользящего окна
$\mathbf{x}_i$	d-мерный вектор наблюдений в момент времени t_i
X_k	матрица скользящего окна размерности $L \times d$
y_k	целевая переменная (значение $common\_event\_delay$)
$\mathcal{N}$	обучающая выборка
N	общее количество обучающих примеров
f^*	неизвестная целевая функция
A	разрабатываемый алгоритм прогнозирования
ε	допустимая погрешность прогнозирования
Δ	горизонт прогнозирования (15 секунд)
end-to-end	сквозной (от начала до конца процесса)
inference	процесс получения оценок от обученной модели
warm-start	инициализация обучения с предобученными параметрами