Факультет вычислительной математики и кибернетики МГУ имени М.В. Ломоносова

Программа профессиональной переподготовки «Машинное обучение и управление большими данными»

Сравнительный анализ методов машинного обучения для прогнозирования деградации ИТ инфраструктуры

Фурсикова Вера Владимировна

Цели и задачи

Цель работы

Проведение сравнительного анализа методов ML для прогнозирования деградации ИТ-инфраструктуры

Постановка задачи

По истории метрик машины за последние 30 наблюдений (~ 30 минут) предсказать, перейдёт ли она в состояние деградации в течение следующих 10 наблюдений (HORIZON = 10, ~10 минут)

План работ

1. Анализ подходов
2. Выбор датасета
3. Анализ структуры данных
4. Выбор целевых переменных
5. Выбор классов состояния
6. Выбор метрик
7. Предобработка
8. Реализация и обучение 4 моделей
9. Сравнение по метрикам

Актуальность

Почему задача важна

- Современные ИТ-системы генерируют большие объемы данных мониторинга, ручной анализ невозможен
- Отказы и деградация сервисов приводят к финансовым и репутационным потерям
- Традиционный мониторинг реагирует на проблему после её возникновения
- Существует потребность в предсказании проблем до их возникновения

Преимущества перед традиционным мониторингом

- Прогнозирование деградации вместо констатации сбоя
- Низкий показатель срабатывания на ложные тревоги
- Минимизация затрат за счет предотвращения отказов систем

Источник данных

Данные:

Источник данных: Alibaba Cluster Trace v2018
Датасет: machine_usage.csv
Период сбора данных: 8 дней
Количество серверов: 4023
Количество записей: >246 млн

Описание:

Alibaba Cluster Trace v2018 - открытый набор данных Alibaba. Он содержит детализированные логи работы реального производственного кластера и широко используется исследователями для изучения характеристик облачных систем, планирования ресурсов и оценки алгоритмов

EDA для machine_usage.csv

Сводная статистика по числовым колонкам

	dtype	missing_count	missing_pct	mean	std	min	max
cpu_util_percent	float32	6	0.000	38.166	16.053	0.0	100.0
mem_util_percent	float32	0	0.000	88.207	13.875	0.0	98.0
mem_gps	float32	195912868	79.338	3.169	3.014	0.0	100.0
mkpi	float32	195912868	79.338	0.599	0.577	0.0	18.0
net_in	float32	0	0.000	41.471	10.524	0.0	100.0
net_out	float32	0	0.000	32.874	9.216	0.0	100.0
disk_io_percent	float32	0	0.000	7.844	8.459	-1.0	101.0

Доля строк выше порогов

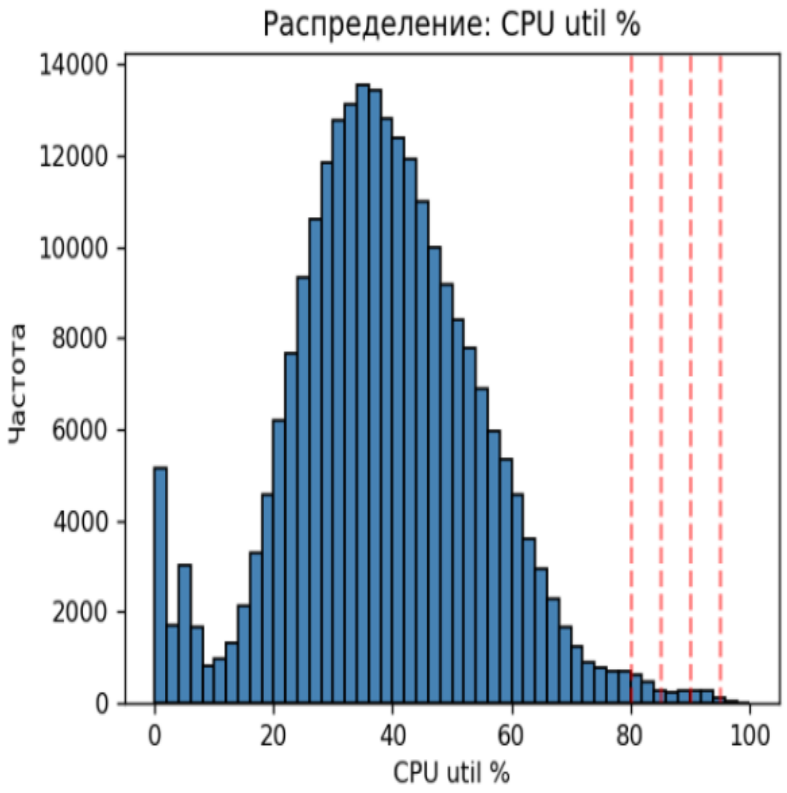
	>= 80%	>= 85%	>= 90%	>= 95%
cpu_util_percent	1.0965	0.5901	0.2990	0.0456
mem_util_percent	94.2125	87.3871	63.7583	15.8739
disk_io_percent	0.2402	0.1985	0.1685	0.1420

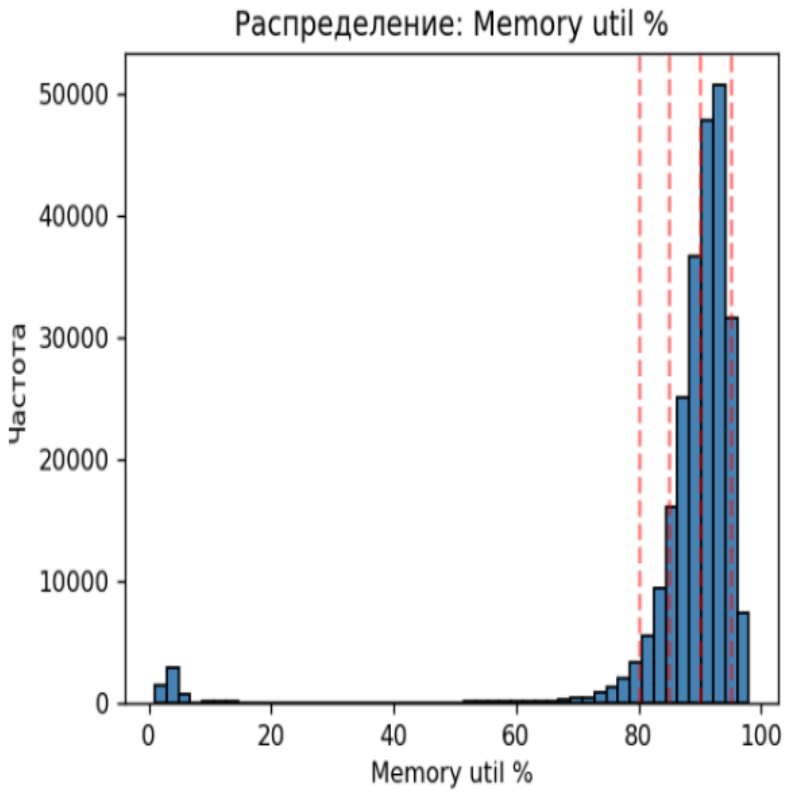
Количество уникальных машин: 4,023
Всего строк: 246,934,820
Пропуски по machine_id/timestamp: 0, 0

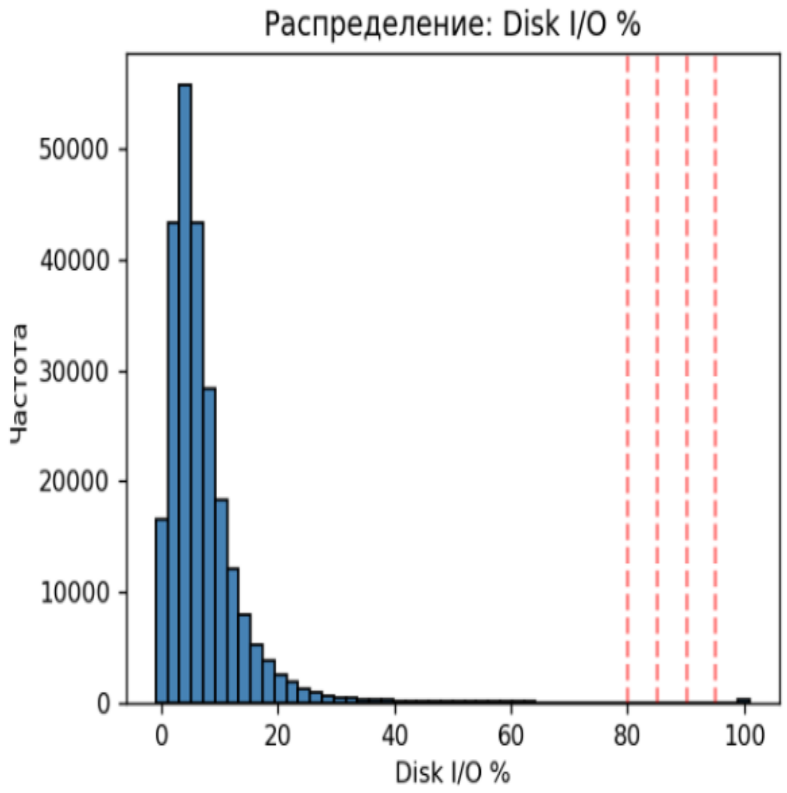
Интервал между наблюдениями (сек)
медиана: 60.0
среднее: 99.8
мода: 10.0

EDA для machine_usage.csv Графики

Распределение значений







Выводы из EDA

CPU util

Данные показывают, что для cpu_util преобладают значения от 20 до 60%, что является довольно стандатным показателем для отрасли. Значения выше 80% встречаются примерно в 1% случаев. Параметр хорошо подходит для обучения.

Memory util

Данные показывают, что для memory_util преобладают значения выше 80%. 94% всех значений превышает порог в 80% и 15% всех значений превышают порог в 95%, что является крайне высокой утилизацией. Однако, для данного кластера, судя по распределению это норма, что может говорить о разном. Как о высокой стабильности системы и возможности позволить себе столь высокую утилизацию на постоянной основе, так и о неэффективном управлении памятью. Параметр не подходит для обучения, поскольку память всегда загружена на высоком уровне, её абсолютные значения малоинформативны для прогнозирования — в отличие от CPU

Disk I\O

Дисковая подсистема имеет большой запас производительности. Низкое среднее значение при стандартном отклонении 8.46% может указывать на то, что большую часть времени диск не используется и нагрузка возникает всплесками. В данной задаче не будет использоваться для обучения.

Постановка задачи классификации

Формулировка:

По временному ряду загрузки CPU за последние **30 измерений** (~30 минут) необходимо предсказать, в какое из четырёх состояний перейдёт система через **10 измерений** (~10 минут).

Классы состояния:

Класс	Название	CPU	Доля в данных
0	Normal	<85%	99.54%
1	Warning	85-90%	0.26%
2	Alarm	90-95%	0.18%
3	Critical	>95%	0.023%

Подготовка данных

Партицирование по machine_id

Для корректной разметки нужны полные временные ряды по каждой машине, отсортированные по timestamp. Так как каждый раз сортировать такой объем заново трудозатратно, данные сортируются отдельным скриптом и сохраняются в отдельных файлах для каждой машины.

Разметка по CPU с горизонтом предсказания

HORIZON = 10 (~10 минут вперёд при медианном интервале 60 сек)

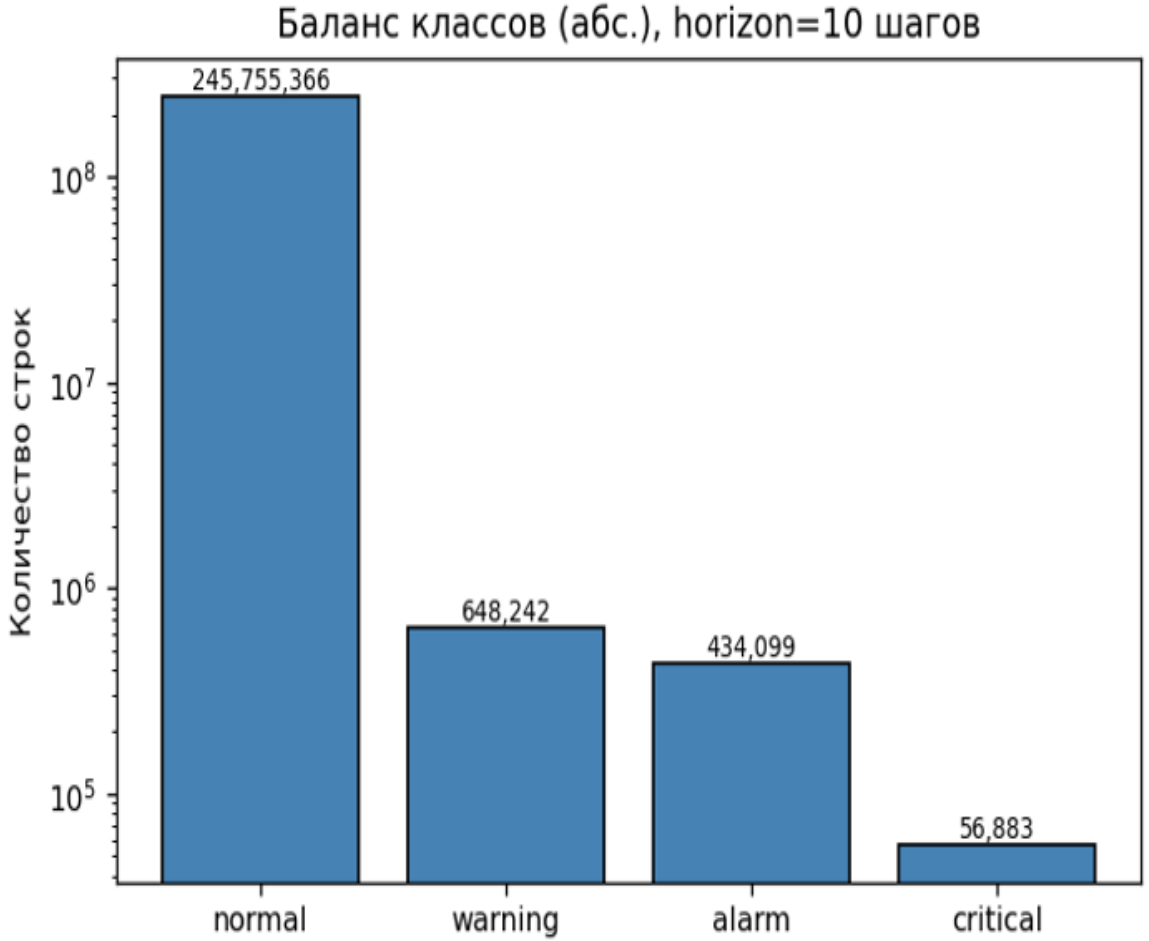
В файл для каждой машины добавляется новая колонка `label`, такая что:

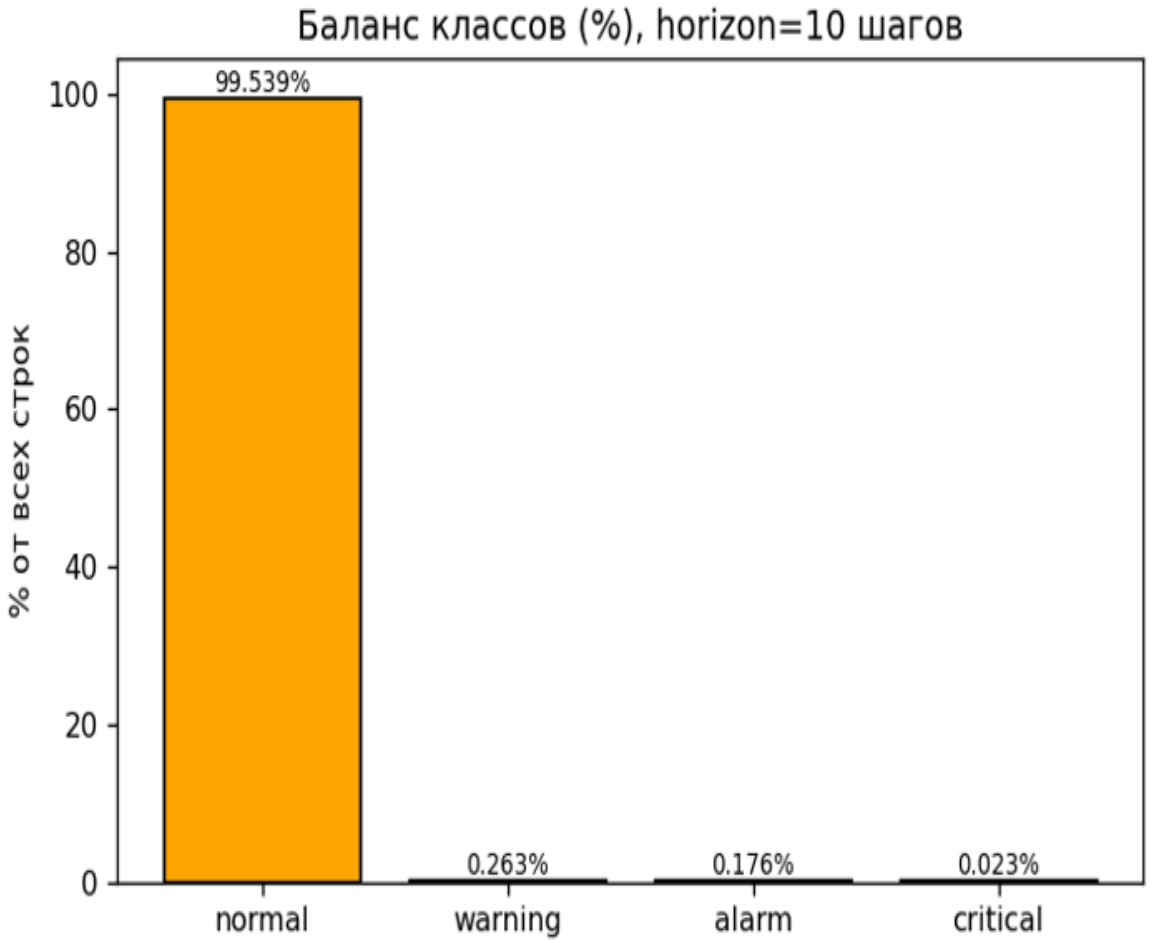
- Метка строки i = класс, который будет актуален в момент i + HORIZON.
- Последние HORIZON (10) строк каждой машины удаляются, так как у них нет метки.

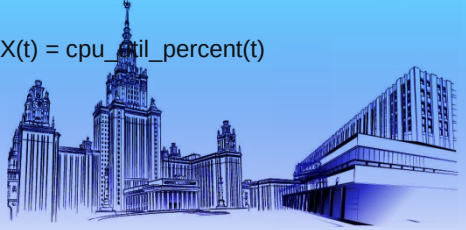
```
Всего строк после разметки: 246,894,590
(каждая машина потеряла последние 10 строк – у них нет метки будущего)

Баланс классов
 class_id class_name     count     pct
        0     normal 245755366 99.5386
        1    warning    648242  0.2626
        2      alarm    434099  0.1758
        3   critical     56883  0.0230
```

Баланс классов после подготовки данных







Создание признаков

Одно лишь текущее значение загрузки процессора не позволяет судить о том, развивается ли в системе деградация. Кратковременный скачок CPU до 95% может быть вызван штатной фоновой задачей и не представляет угрозы, тогда как устойчивый рост нагрузки с 40% до 80% на протяжении получаса может быть ранним признаком приближающегося сбоя. Пороговый подход учитывает только текущее значение и не видит, как система вела себя в последнее время, а ведь именно характер изменения нагрузки даёт наиболее полную картину её состояния. Поэтому, в дополнение к признаку cpu_util_percent (Xt=cpu_util_percent(t)) были добавлены дополнительные признаки:

Признак	Что показывает
cpu_util_percent_roll_mean_30	Сглаженный уровень (среднее за 30 наблюдений). roll_mean_30(t) = (1/30) * Σ X(t-i), где i = 0..29
cpu_util_percent_roll_std_30	Скользящее стандартное отклонение за 30 наблюдений. roll_std_30(t) = √((1/29) * Σ (X(t-i) - roll_mean_30(t))²), где i = 0..29
cpu_util_percent_diff_1	Мгновенная скорость изменения. diff_1(t) = X(t) - X(t-1)
cpu_util_percent_trend_30	Тренд за 30 наблюдений. trend_30(t) = X(t) - X(t-30)

Разделение на Train и Test

Логика

1. Для каждой машины читаем только колонку `label`, считаем количество строк класса 1 (warning), 2 (alarm), 3 (critical) и суммарно n_rare = n1+n2+n3

2. Сортируем машины по убыванию n_rare. Идём по списку и кладём каждую машину в train или test - туда, где сейчас больше не хватает редких строк относительно цели. Машины с n_rare == 0 распределяются тем же способом в конце

3. Для каждой машины читаем полные данные, сортируем по timestamp, применяем сэмплирование normal-класса и добавляем в train/test. Оставляем только 20% от исходного количества строк normal-класса

Результат

```
Train: 22,696,885
Test:  27,596,788

Баланс классов в Train
class 0: 21,784,878 (95.982%)
class 1:    466,828 (2.057%)
class 2:    393,515 (1.734%)
class 3:     51,664 (0.228%)

Баланс классов в Test
class 0: 27,369,571 (99.177%)
class 1:    181,414 (0.657%)
class 2:     40,584 (0.147%)
class 3:      5,219 (0.019%)
```

Выбранные модели

Модель	Тип	Описание	Особенности
Логистическая регрессия	**Линейная**	Линейная модель классификации. Оценивает вероятность принадлежности объекта к классу через линейную комбинацию признаков + сигмоидную функцию	Простота, интерпретируемость, высокая скорость
HistGradientBoosting	**Бустинг**	Градиентный бустинг на деревьях решений. Последовательно строит деревья, где каждое следующее исправляет ошибки предыдущих. Использует гистограммный подход - непрерывные признаки разбиваются на бины (по умолчанию 255), что ускоряет обучение	Быстрее классического градиентного бустинга за счёт дискретизации признаков
XGBoost	**Бустинг**	Градиентный бустинг с регуляризацией. Последовательно строит деревья, каждое следующее корректирует ошибки предыдущих. Отличается встроенной L1/L2-регуляризацией, которая контролирует сложность модели и снижает переобучение	Высокая точность, устойчивость к шуму, поддержка многопоточности, работа с пропусками
LightGBM	**Бустинг**	Градиентный бустинг с гистограммным подходом. Как и HGB, использует гистограммы, но с двумя ключевыми инновациями: Gradient-based One-Side Sampling - оставляет градиенты с большими ошибками и случайно выбирает остальные, и Exclusive Feature Bundling - связывает взаимоисключающие признаки	Leaf-wise стратегия роста деревьев вместо level-wise - дерево растёт в глубину, выбирая лист с максимальным приростом качества. Это быстрее, но требует осторожной настройки (риск переобучения)

Метрики оценки качества

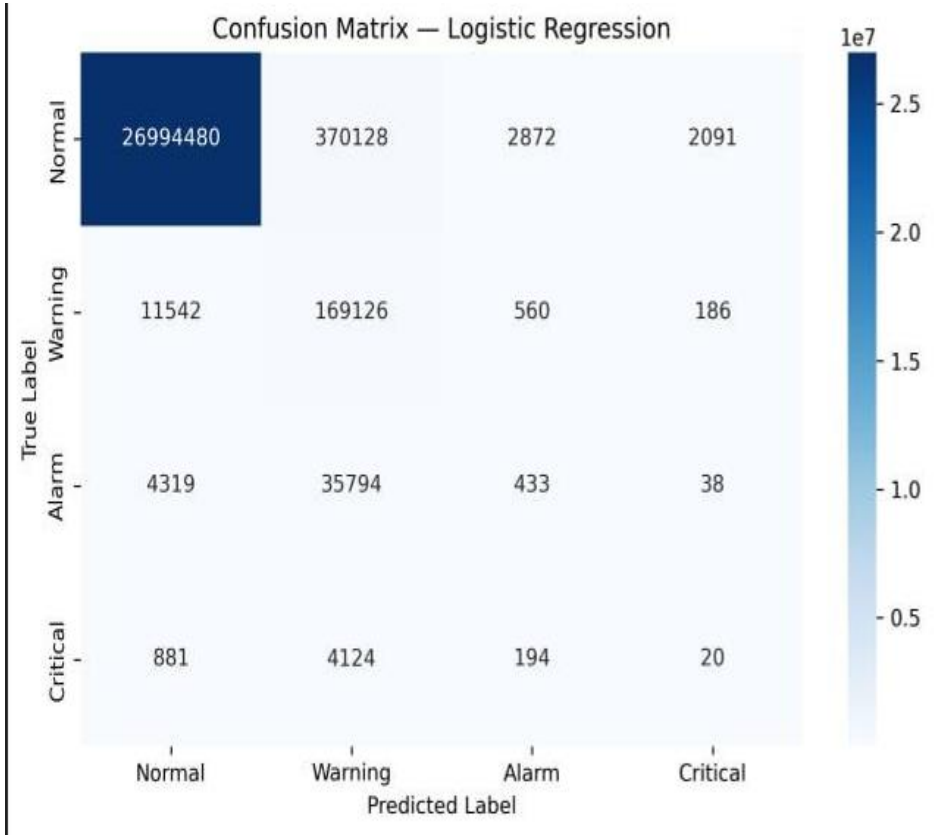
Метрика	Описание	Почему выбрана
F1-macro	Среднее арифметическое F1-мер по всем классам	Даёт равный вес каждому классу, независимо от его размера. Если модель плохо работает на редком классе, это сразу отразится на метрике
F1 по каждому классу	Метрика F1, посчитанная отдельно по каждому классу	Показывает с каким классом модель хуже работает
Recall по редким классам	Метрика, показывающая долю правильно найденных аномалий среди всех реальных аномалий	В задаче прогнозирования отказов пропустить аномалию (FN) гораздо дороже, чем ошибочно объявить тревогу (FP). Recall показывает, сколько реальных проблем мы нашли
Confusion Matrix	Таблица, где строки - реальные классы, столбцы - предсказанные	Позволяет визуально увидеть, когда модель путает классы, и выявить систематические ошибки

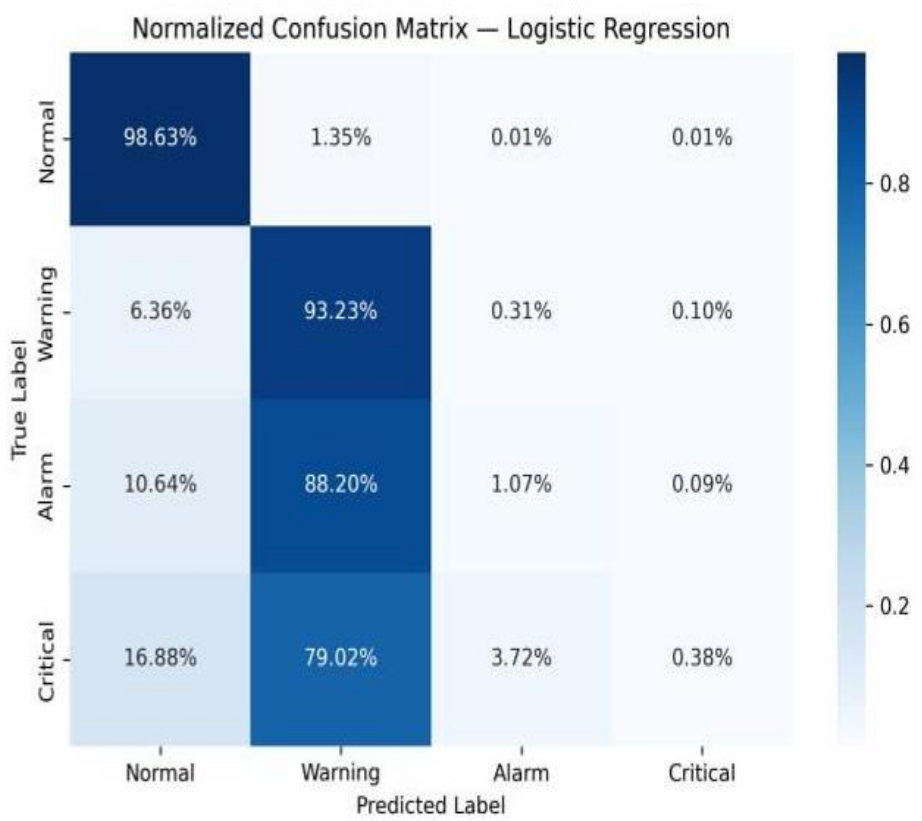
Результат модели «Логистическая регрессия»

F1-macro: 0.3656

F1 по классам:
 Normal: 0.9928
 Warning: 0.4447
 Alarm: 0.0194
 Critical: 0.0053

Recall по редким классам:
 Warning: 0.9323
 Alarm: 0.0107
 Critical: 0.0038



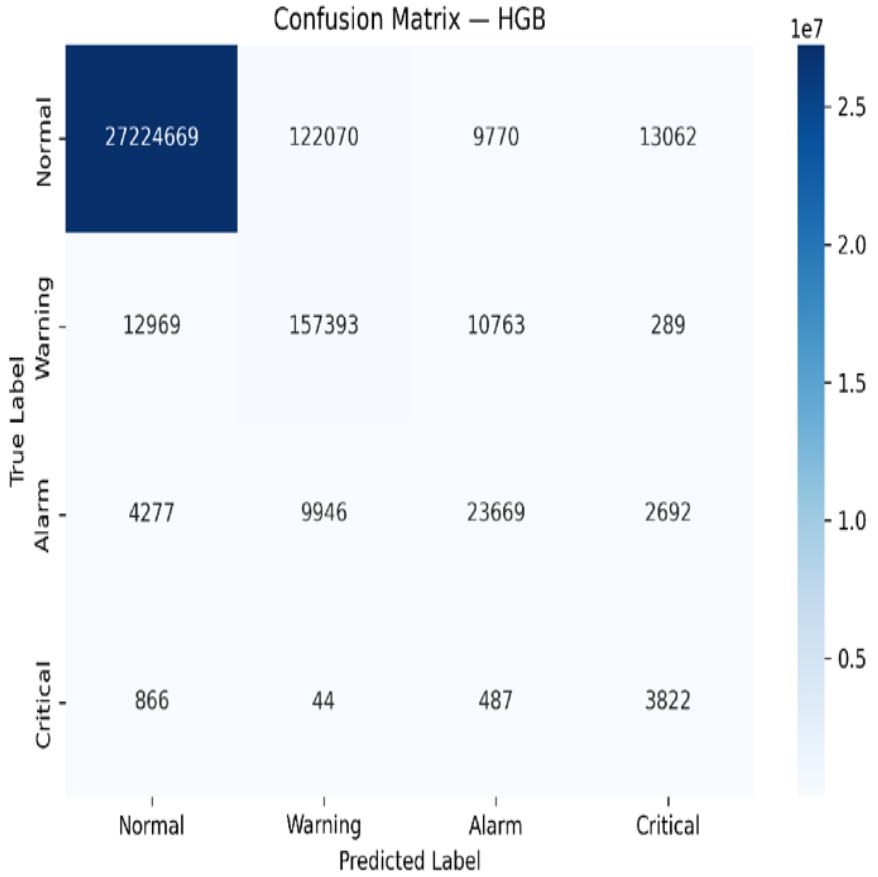


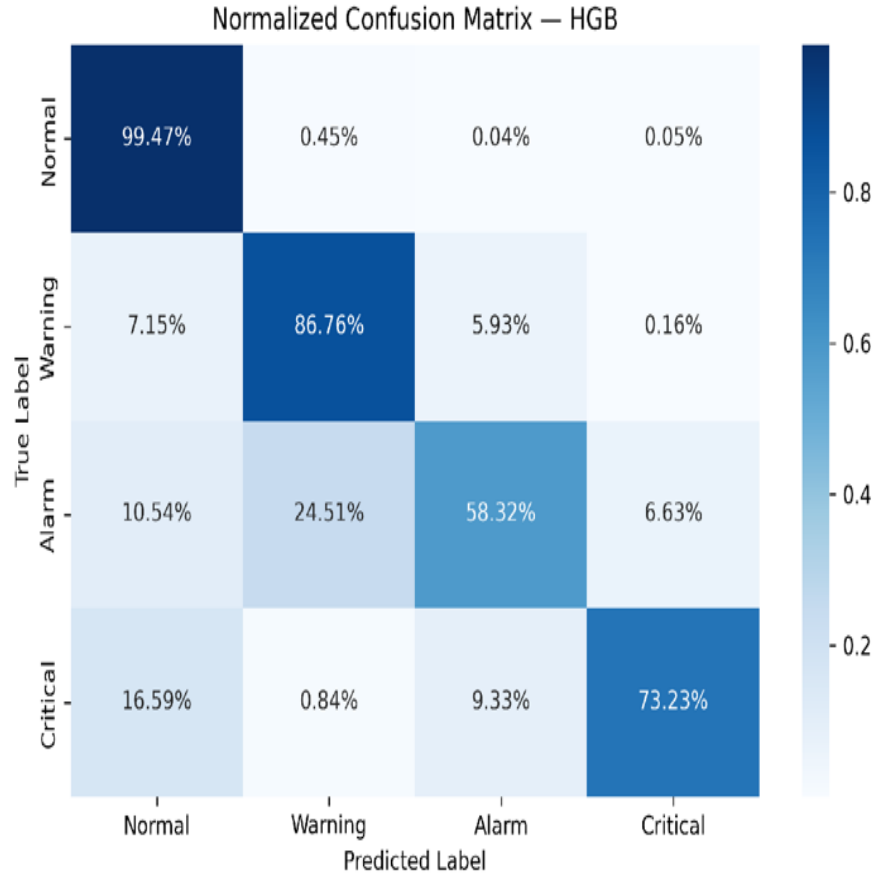
Результат модели «HistGradientBoosting»

F1-macro: 0.6314

F1 по классам:
 Normal: 0.9970
 Warning: 0.6685
 Alarm: 0.5551
 Critical: 0.3047

Recall по редким классам:
 Warning: 0.8676
 Alarm: 0.5832
 Critical: 0.7323



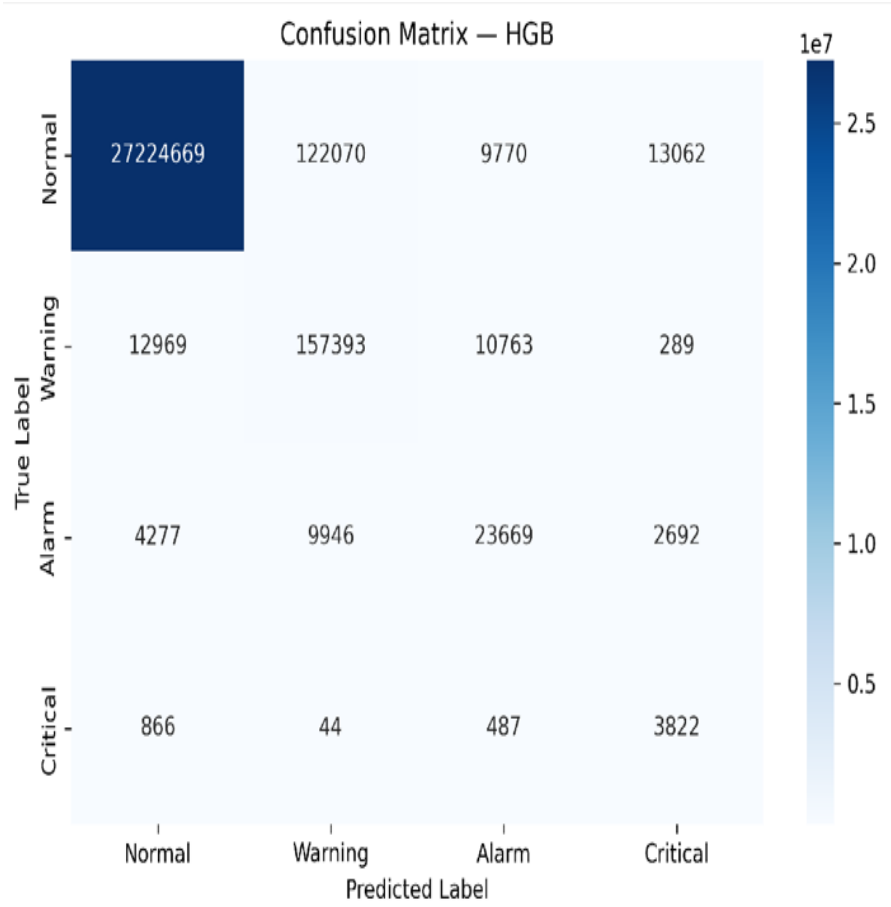


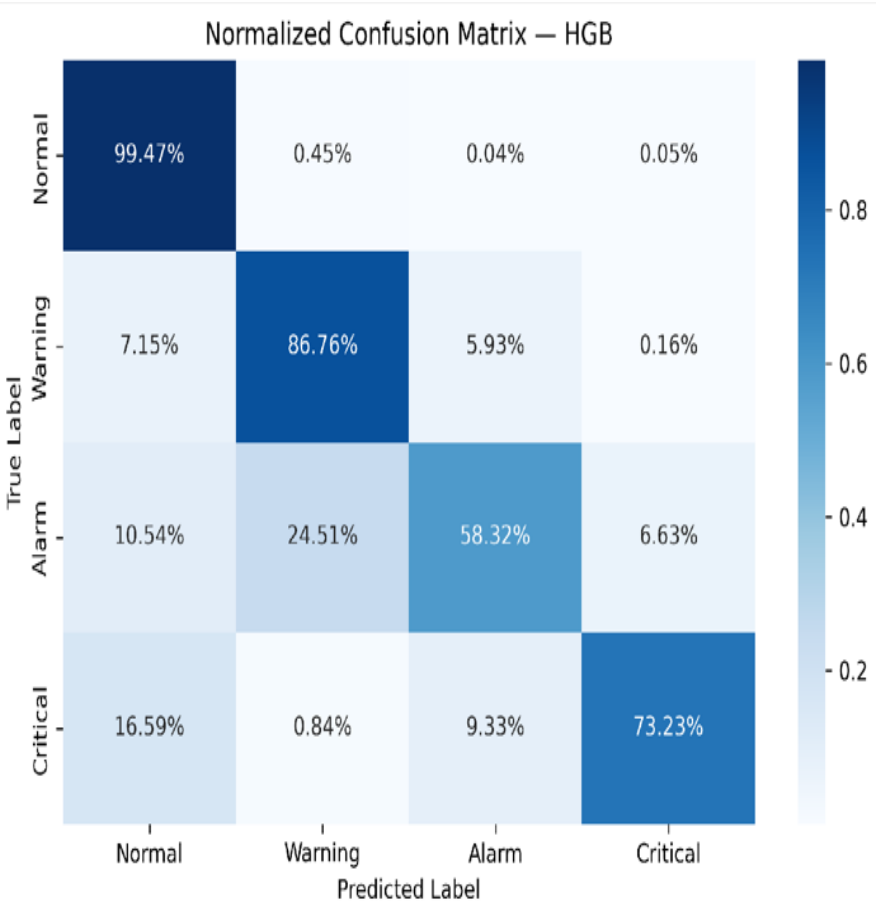
Результат модели «XGBoost»

F1-macro: 0.7077

F1 по классам:
 Normal: 0.9975
 Warning: 0.6779
 Alarm: 0.6082
 Critical: 0.5472

Recall по редким классам:
 Warning: 0.8659
 Alarm: 0.6050
 Critical: 0.6754



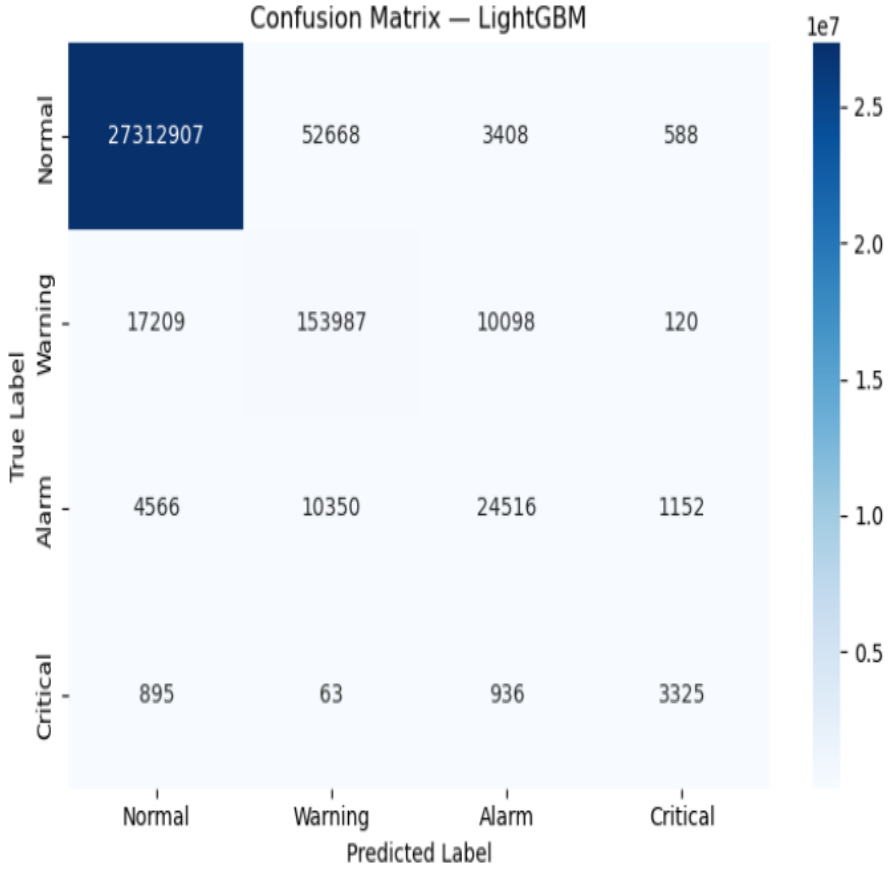


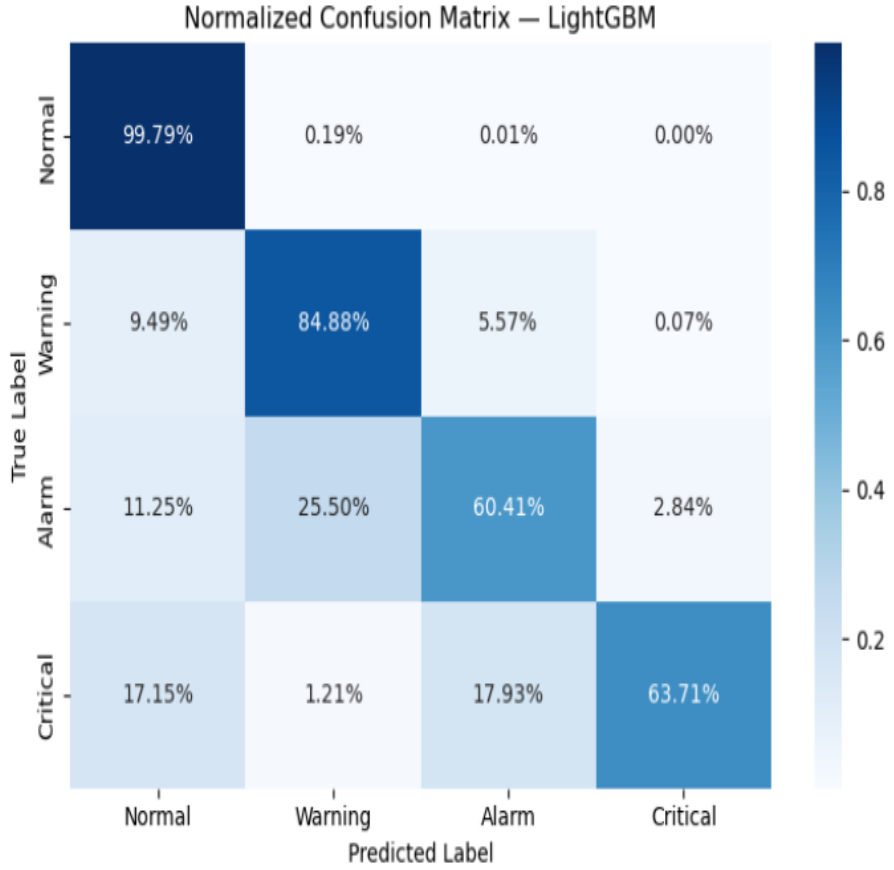
Результат модели «LightGBM»

F1-macro: 0.7568

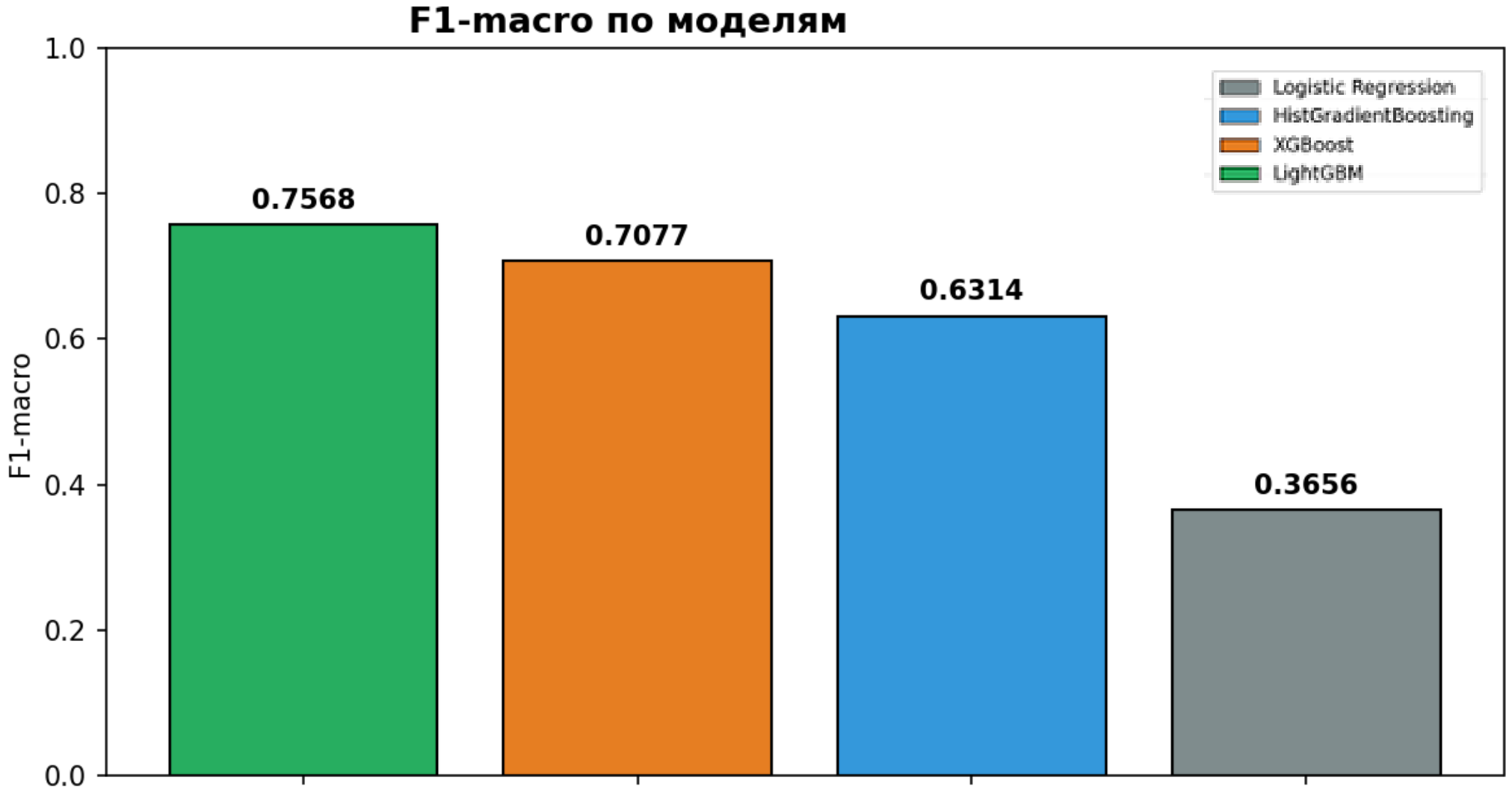
F1 по классам:
 Normal: 0.9985
 Warning: 0.7729
 Alarm: 0.6164
 Critical: 0.6392

Recall по редким классам:
 Warning: 0.8488
 Alarm: 0.6041
 Critical: 0.6371

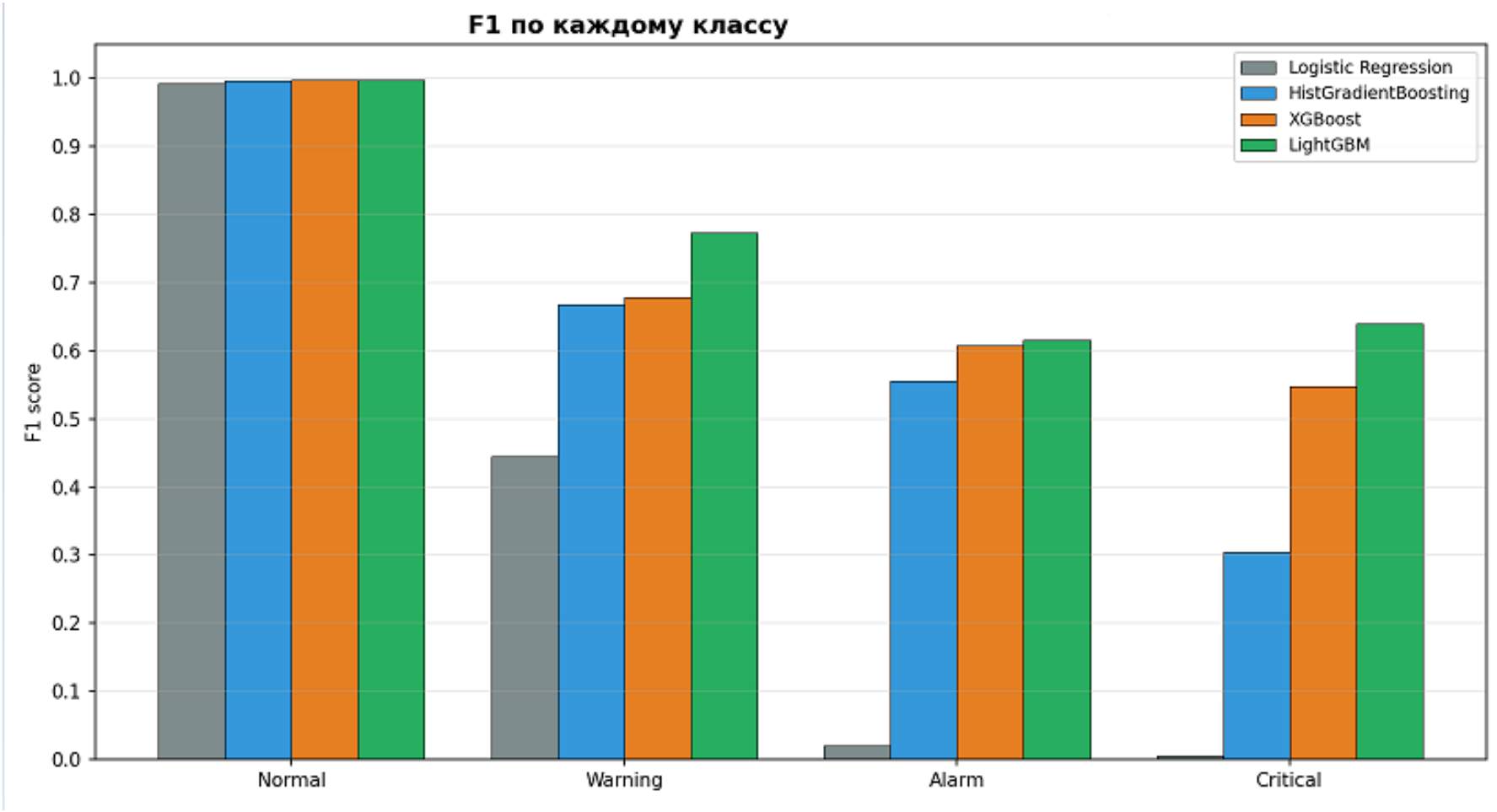




Сравнение моделей. F1 macro

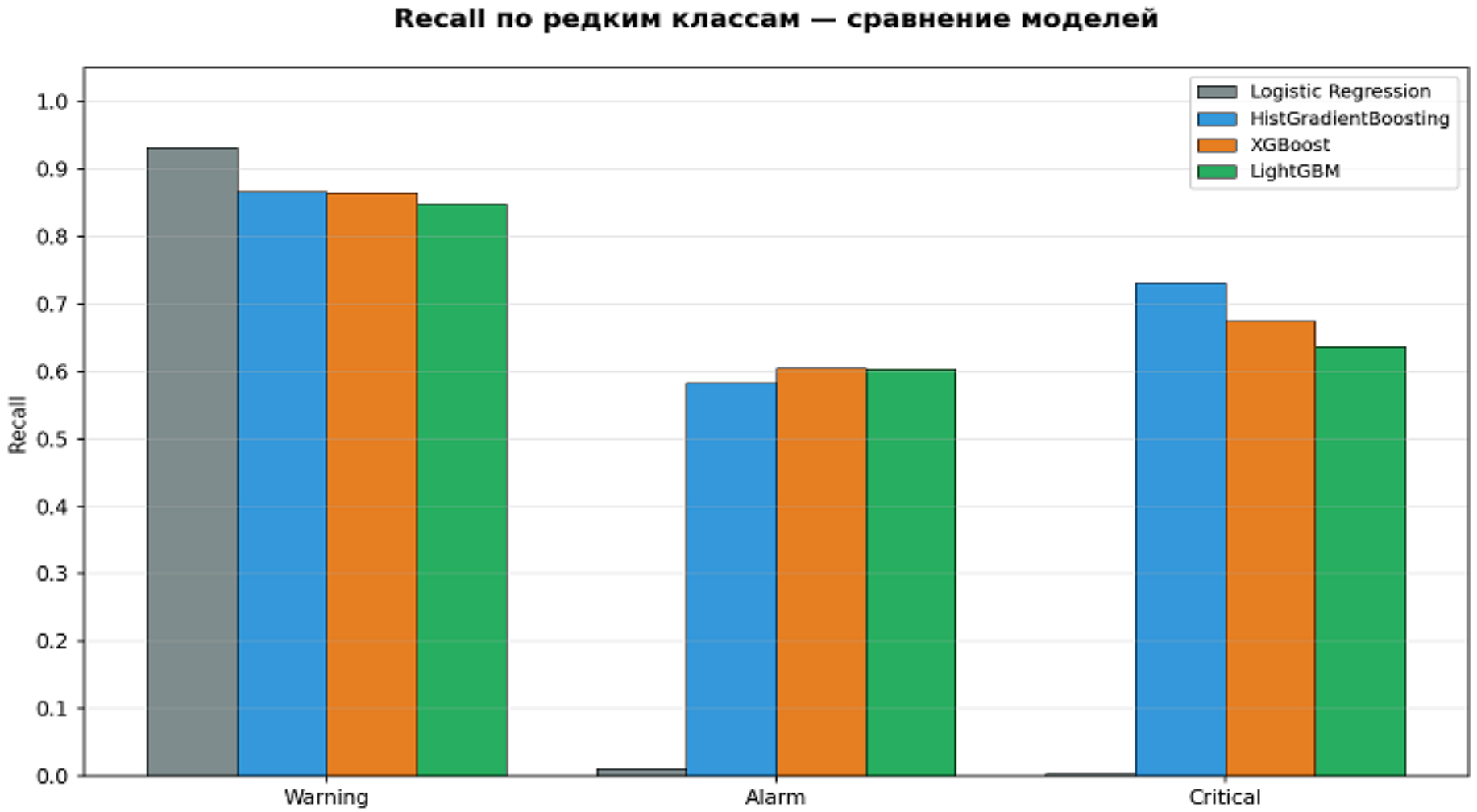


Сравнение моделей. F1 по каждому классу

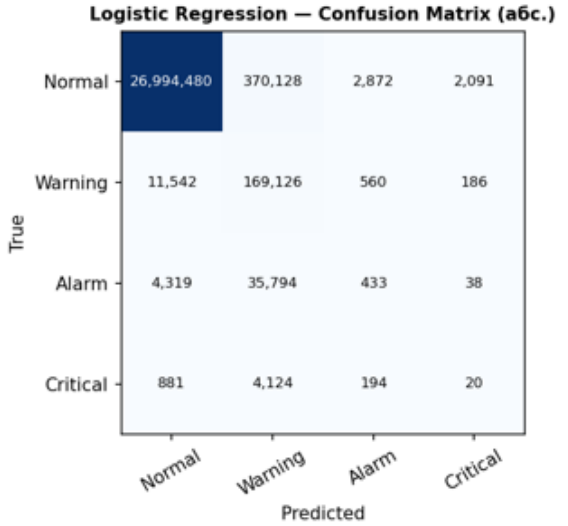


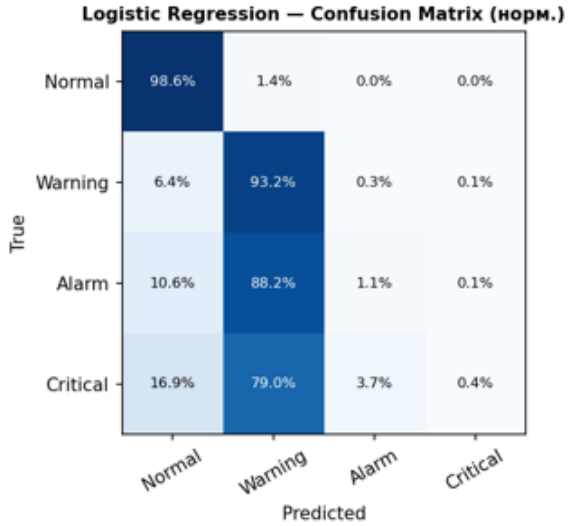
Сравнение моделей. F1 по классам. Сильные\слабые стороны

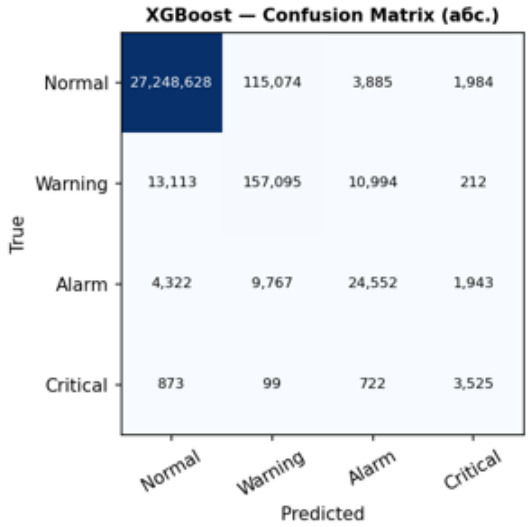
Сравнение моделей. Recall по редким классам

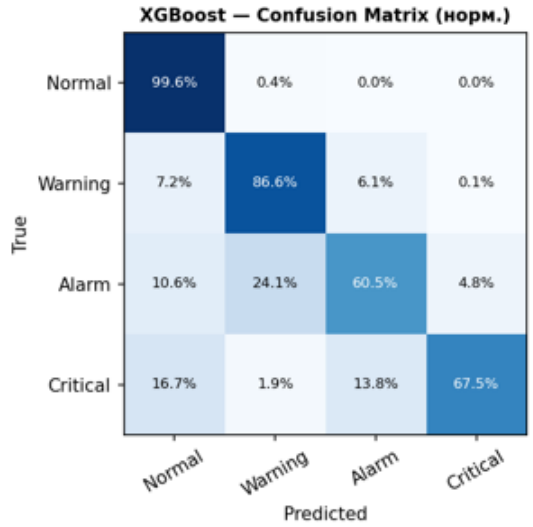


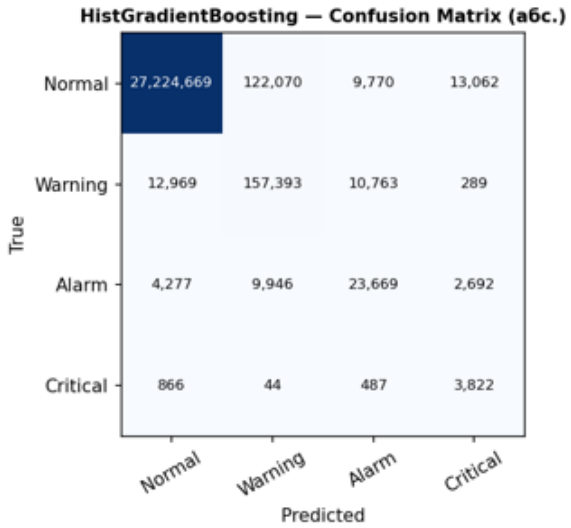
Сравнение моделей. Confusion Matrix

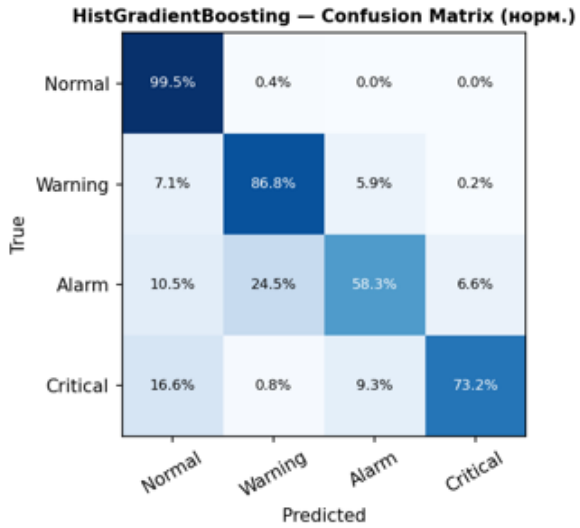


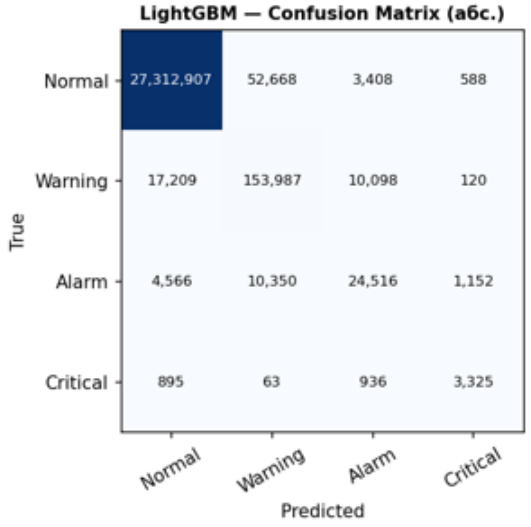


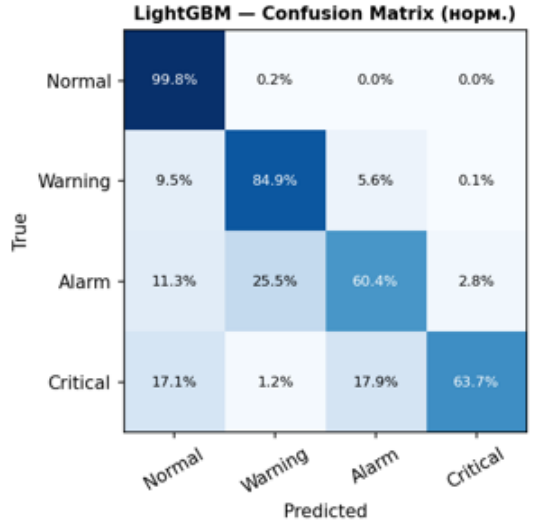












Выводы и перспективы развития

Вывод

Проведён сравнительный анализ четырёх моделей для прогнозирования деградации ИТ-инфраструктуры по данным Alibaba Cluster Trace.

LightGBM показал лучшее сбалансированное качество, HistGradientBoosting - лучшую полноту на критических состояниях. Логистическая регрессия неприменима из-за сильного дисбаланса классов.

Основная ошибка всех моделей - неверное распознавание Alarm и Warning.

Таким образом, LightGBM рекомендуется как основная модель благодаря лучшему сбалансированному качеству и небольшому лидированию по остальным метрикам.

Перспективы развития

- Расширение признакового пространства - включение метрик памяти, диска и сети. Это может повысить качество на других типах данных, где деградация связана с другими ресурсами
- Применение LSTM - рекуррентные нейросети способны самостоятельно извлекать паттерны из сырых временных рядов без ручного конструирования признаков, что потенциально может дать прирост качества.
- Интеграция в реальном времени с Prometheus/Zabbix

Список литературы

- Машинное обучение и управление большими данными: курс лекций. Факультет вычислительной математики и кибернетики МГУ имени М.В. Ломоносова, 2026.
- А. Чеснов, К. Нагорный, Т. Чирков. Эксплуатация ЦОД. Практическое руководство». - М. Изд-во Aegitas, 2024. - 442 с. -
- XGBoost Documentation [Электронный ресурс] - URL: https://xgboost.readthedocs.io/
- LightGBM Documentation [Электронный ресурс] - URL: https://lightgbm.readthedocs.io/
- TensorFlow Documentation [Электронный ресурс] - URL: https://www.tensorflow.org/
- Pandas: Python Data Analysis Library [Электронный ресурс] - URL: https://pandas.pydata.org/docs/
- NumPy: Fundamental package for scientific computing in Python [Электронный ресурс] - URL: https://numpy.org/doc/stable/
- Alibaba Cluster Data [Электронный ресурс] - URL: clusterdata/cluster-trace-v2018 at master · alibaba/clusterdata · GitHub