Дюсенов Дамир Сергеевич, магистрант
Российский новый университет
Dyusenov Damir Sergeevich, Master's student
Russian New University

Трефилова Ольга Леонидовна,
старший преподаватель кафедры информационных систем
в экономике и управлении, Российский новый университет
Trefilova Olga Leonidovna, Senior Lecturer at the Department
of Information Systems in Economics and Management,
Russian New University

МЕТОД ВЫЯВЛЕНИЯ АНОМАЛЬНЫХ СОБЫТИЙ В ЖУРНАЛАХ CI/CD НА ОСНОВЕ АЛГОРИТМА ISOLATION FOREST
A METHOD FOR DETECTING ANOMALOUS EVENTS IN CI/CD LOGS USING THE ISOLATION FOREST ALGORITHM

Аннотация. Рассматривается метод автоматизированного выявления аномальных событий в журналах непрерывной интеграции и доставки программного обеспечения. Предложены состав признаков журнальной записи, порядок подготовки данных и программная реализация алгоритма Isolation Forest на языке Python. Описано встраивание разработанного модуля в DevOps-портал и определены ограничения его практического применения.

Abstract. The paper considers a method for automated detection of anomalous events in continuous integration and delivery logs. A log feature set, a data preparation procedure, and a Python implementation of the Isolation Forest algorithm are proposed. The integration of the module into a DevOps portal and the limitations of its practical use are described.

Ключевые слова: CI/CD, анализ журналов, обнаружение аномалий, Isolation Forest, DevOps, машинное обучение.

Keywords: CI/CD, log analysis, anomaly detection, Isolation Forest, DevOps, machine learning.

Введение

Непрерывная интеграция и доставка программного обеспечения формируют поток технических сообщений о сборках, тестировании, взаимодействии сервисов и развертывании на стендах. При большом количестве заданий ручной просмотр журналов становится узким местом: специалисту приходится сопоставлять тысячи строк, а значимое отклонение может не содержать явного признака ERROR. Задержка обнаружения увеличивает время восстановления и создает риск переноса дефекта на следующий этап конвейера.

Распространенный подход основан на словарях ключевых слов и заранее заданных порогах. Он эффективен для известных ошибок, но плохо выявляет редкие сочетания формально допустимых значений: умеренное увеличение длительности вместе с необычным кодом ответа, нестандартную последовательность этапов либо нетипичную активность на конкретном стенде. В условиях дефицита размеченных примеров целесообразно использовать методы обнаружения выбросов, обучаемые преимущественно на нормальном поведении [1, 2].

Цель статьи – разработать воспроизводимый метод первичной приоритизации событий CI/CD на основе Isolation Forest. Метод не заменяет расследование и не принимает решение о выпуске. Его результатом является оценка аномальности, по которой записи сортируются для последующей проверки DevOps-специалистом.

Формирование признакового описания

Исходная строка журнала содержит временную отметку, идентификатор системы, стенд, этап задания, уровень сообщения, длительность операции, код ответа и текст. До анализа из текста удаляются токены, персональные данные и иные секреты. Категориальные значения кодируются численно, числовые признаки масштабируются, а пропуски обрабатываются единообразно. Версия преобразования фиксируется вместе с версией модели, поскольку изменение кодировщика способно изменить распределение оценок.

Выбранные признаки должны быть интерпретируемыми и доступными в момент принятия решения. В таблице 1 приведен минимальный состав данных. Он позволяет выявлять как одиночные выбросы, так и нетипичные сочетания нескольких характеристик без сохранения полного текста сообщения в обучающей выборке.

Таблица 1

Состав признаков журнального события

Признак	**Назначение**	**Подготовка**
Уровень сообщения	Характеризует серьезность записи	Категориальное кодирование
Длительность	Выявляет замедление этапа	Масштабирование
Код ответа	Отражает результат обращения	Число или категория
Этап конвейера	Учитывает контекст операции	Категориальное кодирование
Стенд	Разделяет разные режимы нагрузки	Категориальное кодирование
Редкость шаблона	Оценивает необычность текста	Частотная оценка

Набор признаков не должен содержать сведения, появляющиеся после подтверждения инцидента: это привело бы к утечке целевой информации. Для пилотного внедрения признаки следует рассчитывать одинаковым программным конвейером при обучении и эксплуатации. Сырые записи целесообразно хранить отдельно с ограниченным сроком и ролевым доступом.

Алгоритм выявления аномалий

Isolation Forest строит ансамбль случайных изолирующих деревьев. Для каждого дерева случайно выбираются признак и точка разделения. Редкий объект обычно отделяется от основной массы за меньшее число разбиений, поэтому его средняя длина пути оказывается короче. Нормализованная оценка вычисляется по средней длине пути во всех деревьях; чем выше оценка, тем более необычным считается объект [1, 3].

Предлагаемый порядок включает шесть операций: получение и очистку записи, вычисление признаков, загрузку зафиксированной версии модели, расчет оценки, сравнение с порогом и сохранение результата. Порог определяется ожидаемой долей аномалий contamination, но в промышленной эксплуатации должен настраиваться отдельно для групп систем и стендов. Универсальный порог может создавать чрезмерную нагрузку на специалиста при изменении интенсивности заданий.

Алгоритм реализован на Python в виде класса, который формирует случайные подвыборки, строит деревья, вычисляет среднюю длину изоляции и выбирает порог по обучающим оценкам. Фиксация seed обеспечивает повторяемость эксперимента. В отличие от вызова готовой библиотеки, учебная реализация позволяет проследить связь между построением дерева и итоговой оценкой; для промышленного применения ее результат необходимо сопоставить с проверенной библиотечной реализацией [4, 5].

Архитектура программного модуля

Программный модуль отделен от пользовательского интерфейса и источника данных. Такая граница позволяет заменить файловое хранилище на PostgreSQL, а пакетный источник – на API Jenkins или брокер сообщений без изменения математического ядра. Состав компонентов и направление передачи данных представлены на рисунке 1.

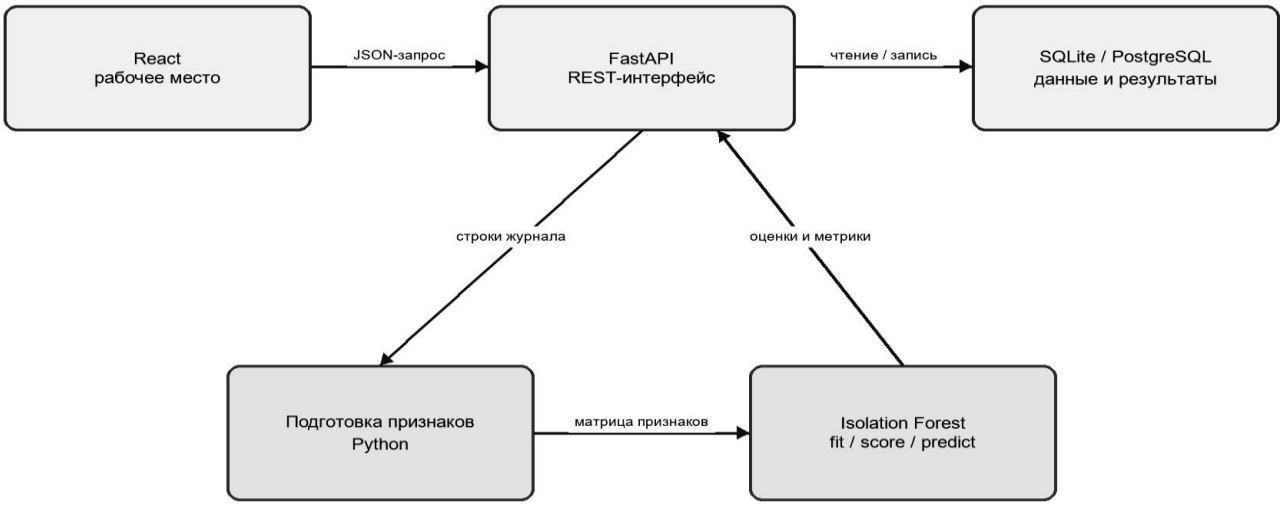

Рисунок 1. Архитектура программного модуля выявления аномалий

На вход конвейера поступают обезличенные события. Модуль подготовки признаков преобразует их в векторы, сервис модели возвращает оценку и признак превышения порога, а прикладной слой сохраняет версию модели и параметры решения. Интерфейс отображает события в порядке убывания оценки. Подтверждение или отклонение сигнала специалистом сохраняется как обратная связь, но не запускает автоматическое переобучение.

Доступ к обучению и изменению порога предоставляется ролям DevOps и администратора. Разработчик или тестировщик просматривает только события разрешенных систем. Каждое решение должно быть воспроизводимым: вместе с ним сохраняются идентификаторы модели и преобразователя признаков, порог, время расчета и исходный контекст. Эти сведения позволяют отличить изменение поведения системы от изменения самой модели.

Эксплуатационные ограничения

Статистическая необычность не доказывает наличие программного дефекта. Новый штатный сценарий также может получить высокую оценку, а последовательность обычных записей – оказаться признаком инцидента, который модель отдельных событий не распознает. Поэтому результат следует формулировать как 'требует внимания', а не как 'обнаружена ошибка'. Автоматическая блокировка выпуска на основании единственной оценки не допускается.

Качество зависит от репрезентативности обучающих данных. После изменения Jenkinsfile, инфраструктуры или профиля нагрузки распределение признаков может сместиться. Необходимо контролировать долю сигналов, оценки подтвержденных событий и расхождение текущего распределения с обучающим. При обнаружении дрейфа модель сначала проверяется в теневом режиме, после чего ответственное лицо утверждает новую версию.

Еще одно ограничение связано с защитой информации. Журналы могут содержать адреса, имена пользователей, параметры запросов и секреты. Маскирование должно выполняться до передачи данных в анализатор, а доступ к исходным строкам – регистрироваться. Требования к хранению, передаче и регистрации доступа целесообразно согласовывать с рекомендациями по управлению компьютерными журналами [6].

Заключение

Предложен метод выявления аномальных событий в журналах CI/CD, включающий очистку данных, формирование интерпретируемых признаков, расчет оценки Isolation Forest и подтверждение результата специалистом. Разработанная архитектура отделяет источник, подготовку признаков, модель, хранилище и интерфейс, что обеспечивает заменяемость компонентов и воспроизводимость решений.

Практическое применение метода следует начинать с теневого пилота на одной системе и некритичном стенде. Перспективы развития связаны с обработкой последовательностей событий, адаптивными порогами для разных стендов, контролем дрейфа и сравнением с альтернативными методами обнаружения выбросов.

Список литературы:

1. Liu F. T., Ting K. M., Zhou Z.-H. Isolation Forest // 2008 IEEE International Conference on Data Mining. Pisa, 2008. P. 413-422. DOI: 10.1109/ICDM.2008.17.

2. Chandola V., Banerjee A., Kumar V. Anomaly Detection: A Survey // ACM Computing Surveys. 2009. Vol. 41, No. 3. Article 15.

3. Liu F. T., Ting K. M., Zhou Z.-H. Isolation-Based Anomaly Detection // ACM Transactions on Knowledge Discovery from Data. 2012. Vol. 6, No. 1. Article 3.

4. Scikit-learn developers. IsolationForest: API documentation. URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html (дата обращения: 17.09.2026).

5. Python Software Foundation. Python 3 documentation. URL: https://docs.python.org/3/ (дата обращения: 17.09.2026).

6. NIST. Guide to Computer Security Log Management: Special Publication 800-92. Gaithersburg, 2006.

7. Jenkins project. Pipeline documentation. URL: https://www.jenkins.io/doc/book/pipeline/ (дата обращения: 17.09.2026)