Санкт-Петербургский государственный университет

Кафедра системного программирования

Классификационные методы обнаружения разладки

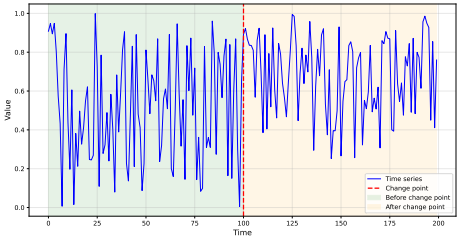
Патов Артемий Сергеевич

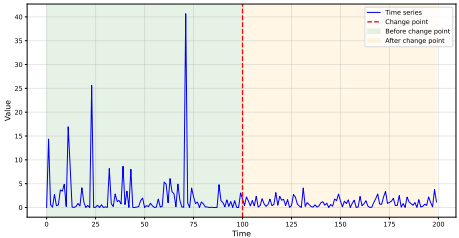
Научный руководитель: доцент кафедры системного программирования, к.ф.-м.н., В. И. Гориховский
Консультант: инженер-исследователь, Лаборатория технологий программирования инфраструктурных решений СПбГУ В. А. Кутуев
Рецензент: Старший инженер, ООО Техкомпания Хуавэй, к.ф.-м.н., М. С. Куля

Санкт-Петербург
2025

Введение

- Разладка (change point) — один из видов аномалий во временных рядах
- Рассматриваются задачи её обнаружения и локализации, которые возникают в различных областях:
  - Алгоритмы контроля нагрузки сети [Li et al., 2021]
  - Бенчмаркинг [Fleming et al., 2023]
- Исследование проводится в рамках проекта PySATL по статистическим алгоритмам
- Предложен новый легковесный подход на основе барьерного метода [Chen, 2019] и классификаторов





Цель исследования

Цель работы — исследование классификационного подхода для решения задачи обнаружения разладки в рамках проекта `pysatl-cpd`.

Основные задачи:

- Разработка обобщенного классификационного барьерного метода обнаружения разладки
- Реализация алгоритмов, основанных на обобщенном классификационном подходе, с использованием классификаторов KNN, Kernel SVM, Decision Tree, Random Forest
- Реализация фреймворка для бенчмаркинга
- Оценка эффективности реализованных алгоритмов с точки зрения статистической мощности и производительности для определенного набора распределений
- Валидация реализованных алгоритмов на реальных данных

Использование офлайн-алгоритма в онлайн-режиме

Классификация алгоритмов

- Офлайн: все данные доступны до начала обработки
- Онлайн: наблюдения поступают во время работы алгоритма

Проблема:

- Требуется онлайн-режим (алгоритм работает в офлайн)
- Множественные точки разладки (алгоритм обрабатывает только одну)

Решение:

- Разделение данных на части, предположение: ≤ 1 точек разладки в каждой части

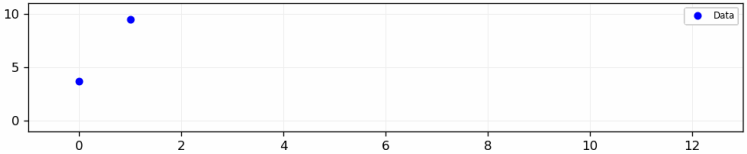
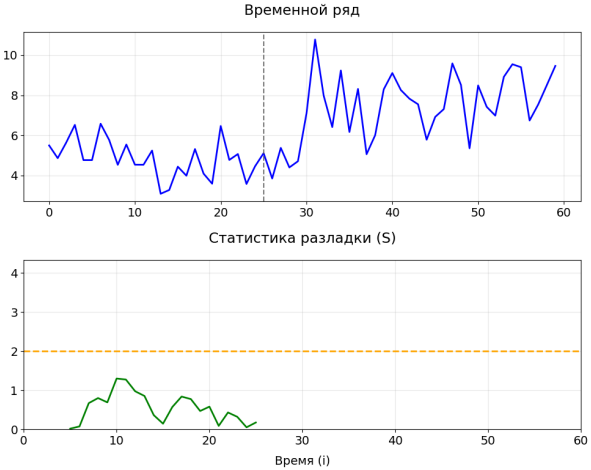


Рис. Алгоритм скрабирования. Размер окна = 4 точки; Сдвиг = 2 точки

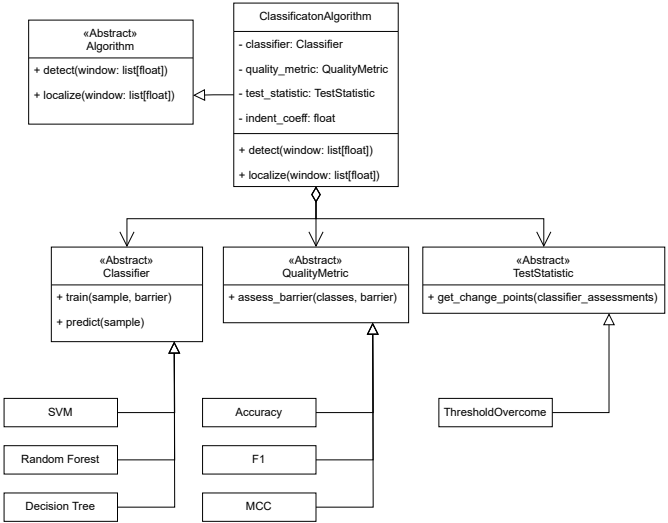
Обобщенный классификационный метод обнаружения разладки

Алгоритм

1. Для i := n_1, ... n_2 вычисляем функцию оценки качества q(i)
   1. Разделяем выборку на два класса: $\zeta = [a_{n_1}, \ldots, a_i]$, $\eta = [a_{i+1}, \ldots, a_{n_2}]$
   2. Строим классификатор на ζ и η
   3. Оцениваем разделимость классов метрикой: q(i)
2. Вычисляем статистику S(q). Например, $\max(q_{n_1}, \ldots, q_{n_2})$, average и т.д.
3. Сравниваем S(q) с порогом



Реализация классификационных методов обнаружения разладки



- Подбор стандартных гиперпараметров алгоритмов на основе Монте-Карло моделирования
- Для автоматической оценки порогов реализован инструмент на основе Монте-Карло моделирования
- Также реализован метод на основе графа kNN [Chen, 2019]

Фреймворк для бенчмаркинга алгоритмов обнаружения разладки

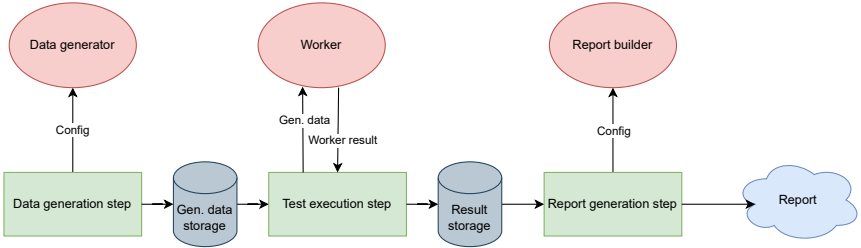
Сценарий оценки алгоритма

1. Генератор: выборки из фиксированных распределений с точкой разладки
2. Воркер: запуск алгоритма, сохранение статистики
3. Отчет: расчет матрицы несоответствий

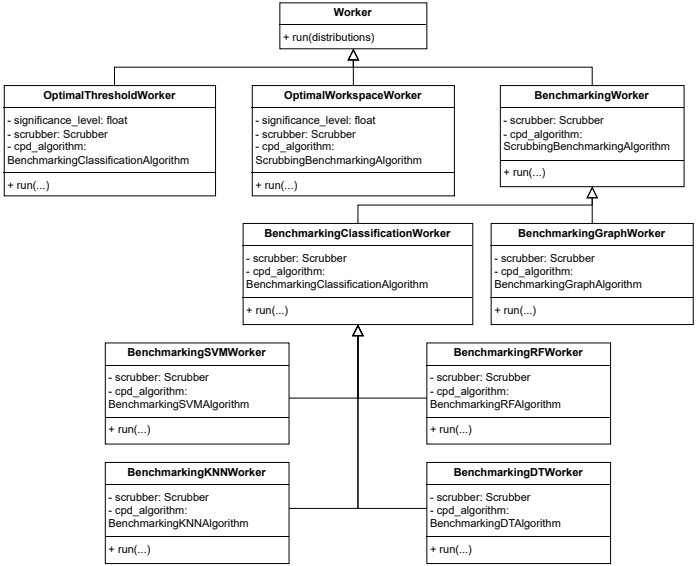
Сценарий оценки порогов

1. Генератор: выборки без точек разладки
2. Воркер: запуск алгоритма, оценка порогов по уровню значимости
3. Отчет: таблица алгоритм-порог

PySATL-experiments

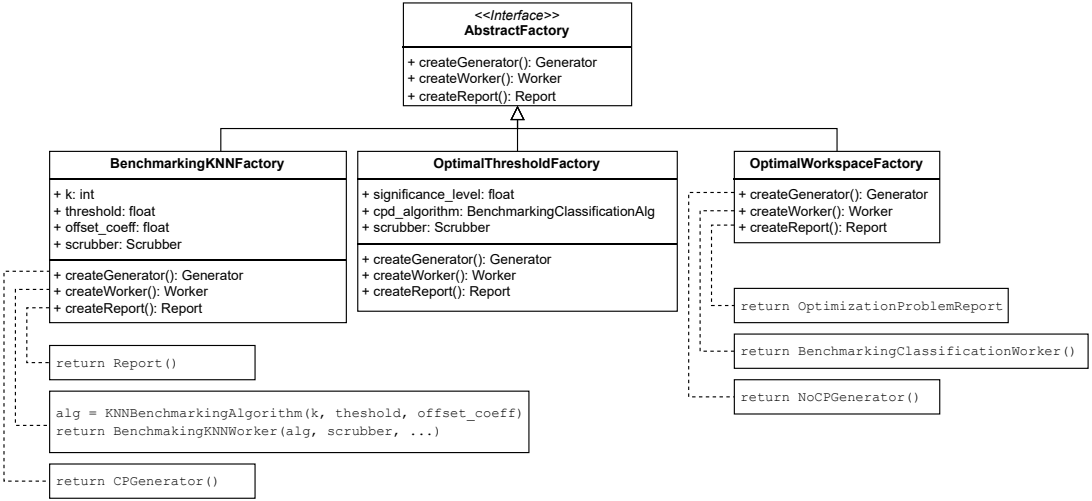


Архитектура воркера



- Древовидная иерархия воркеров обеспечивает возможность сравнения разных алгоритмов по общим метрикам

Согласование компонентов пайплайна под конкретный тип эксперимента



Экспериментальное исследование реализованных алгоритмов

Эксперименты проводились на:

- Аппаратное обеспечение: Intel® Core™ i7-4790 @ 3.60 ГГц (4 ядра, 8 потоков), 32 ГБ RAM
- Программное обеспечение: Ubuntu 22.04, Python 3.12.3

Конфигурация алгоритмов:

- Гиперпараметры по умолчанию
- Пороги для уровня значимости 5%

Данные: комбинации из 15 распределений.

- Для анализа мощности:
  - 10000 выборок каждого распределения
  - 200 наблюдений в выборке
  - Одна точка разладки в центре
- Для производительности:
  - 10000 наблюдений
  - 100 запусков алгоритма

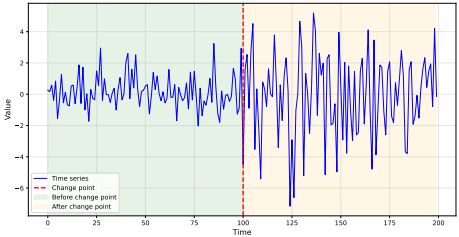


Рис.: Пример выборки. $\mathcal{N}(\mu = 0,\ \sigma = 1)$ меняется на $\mathcal{N}(\mu = 0,\ \sigma = 3)$

Одномерные распределения. Мощность и производительность

	CM	F_1	MCC				
Распр.	7NN graph	7NN	7NN	RF	DT	Lin. SVM	RBF SVM
Beta	0.1681	0.055	0.128	0.1022	0.1017	0.0587	0.1427
Normal_1	0.7858	0.081	0.417	0.3749	0.373	0.2945	0.6974
Uniform_1	0.9003	0.11	0.657	0.4253	0.4227	0.922	0.8617
Uniform_2	0.963	0.124	0.574	0.6572	0.6549	0.55	0.9503
Normal_2	0.9976	0.452	0.94	0.7529	0.7516	0.9898	0.9667
Uniform_3	1.0	0.98	1.0	1.0	1.0	1.0	1.0

Таблица: Мощность, одна точка разладки, $\alpha = 5\%$

- 7NN graph: наилучшая мощность
- SVM: отличные результаты за короткое время
  - Lin. $\rightarrow$ RBF: значительное улучшение
- RF DT, но в 140 раз медленнее
- F_1 значительно хуже MCC
- Время не зависит от числа точек разладки

Алг.\Кол-во разл.	5	10	20
DT	0.7	0.7	0.7
L. SVM	0.8	0.8	0.8
7NN graph	42	42	42
RF	94	94	94

Таблица: Среднее время в с

Многомерные нормальные распределения. Мощность

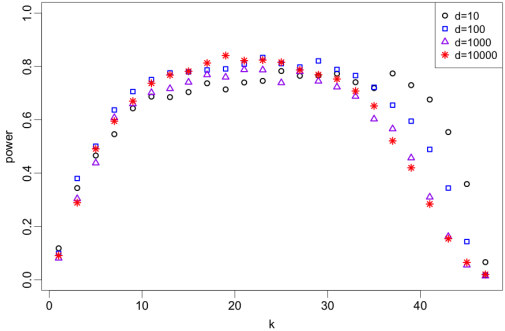


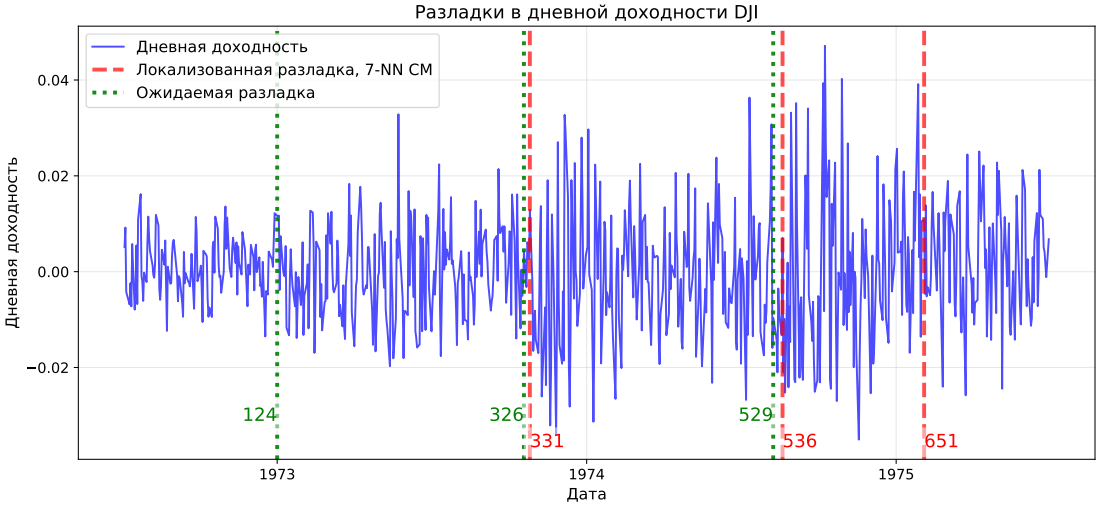
Рис.: Результаты оригинальной реализации [Chen, 2019]. Сдвиги: $\Delta = 1.7$ ($d = 10$), $\Delta = 2.7$ ($d = 100$), $\Delta = 4.5$ ($d = 1000$), и $\Delta = 8$ ($d = 10000$)

	CM	F_1	MCC				
Сдвиг Δ	7NN graph	7NN	7NN	RF	DT	Lin. SVM	RBF SVM
0.7 (d=10)	0.994	0.225	0.841	0.799	0.349	0.757	0.976
1.8 (d=100)	1.0	1.0	1.0	1.0	0.986	1.0	1.0
2.7(d=1000)	1.0	1.0	1.0	1.0	0.999	1.0	1.0

Таблица: Результаты PySATL, $\alpha = 5\%$

- Большинство алгоритмов PySATL превзошли оригинальную реализацию метода на основе kNN

Данные доходности DJI, алгоритм 7-NN CM



Заключение

- Разработан новый подход к обнаружению разладки на основе классификаторов, он представлен на конференции FRUCT 2025[1]
- На основе него реализовано семейство эффективных алгоритмов (`pysatl-cpd`[2], ветка `main`)
- Создан гибкий фреймворк для бенчмаркинга, обеспечивающий надежное сравнение методов (`pysatl-cpd`[2], ветка `scrubbing-methods-bench`)
- Эксперименты на одномерных и многомерных данных показали:
  - Метод на основе kNN чувствителен, но Kernel SVM обеспечивает почти такое же качество при значительно большей скорости
  - Реализация kNN в PySATL превзошла оригинальную [Chen, 2019]
- Реализованные алгоритмы успешно локализовали большинство предполагаемых точек разладки в реальных данных

1) Публикация: `https://doi.org/10.23919/FRUCT65909.2025.11008030` (Дата обращения: 26.05.2025)
2) `pysatl-cpd` (nick: `artemiipatov`): `https://github.com/PySATL/pysatl-cpd` (Дата обращения: 26.05.2025)

Задачи детекции и локализации разладки

- Гипотеза разладки:

$$\{x_i\}, i = 1, \dots, n$$
$$H_0 \colon x_i \sim F_0, i = 1, \dots, n$$
$$H_a \colon \exists 1 \leq \tau < n, x_i \sim \begin{cases} F_1, i > \tau \\ F_0, \text{иначе} \end{cases}$$

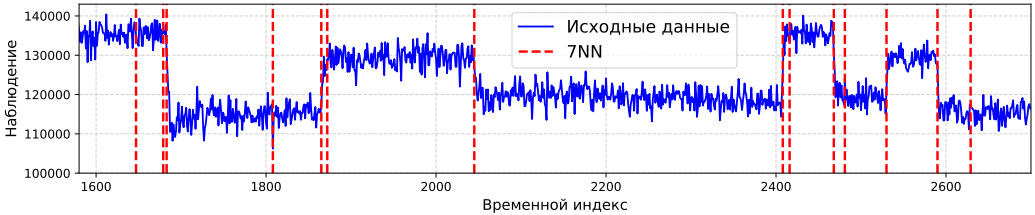
- Гипотеза корректной локализации:

$$\{x_i\}, i = 1, \dots, n$$
$$H_0 \colon x_i \sim F_0, i = 1, \dots, n$$
$$H_a \colon 1 \leq \tau_1 < n \Rightarrow \exists \tau \in [\tau_1 - k; \tau_1 + k], x_i \sim \begin{cases} F_1, i > \tau \\ F_0, \text{иначе} \end{cases}$$

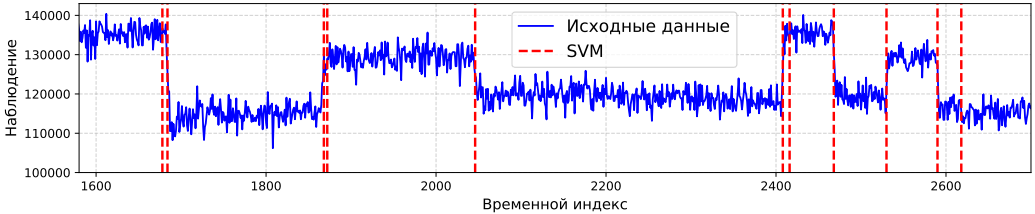
τ_1 — корректно распознанная с точностью k разладка

Реальные данные. Алгоритмы на основе KNN и SVM

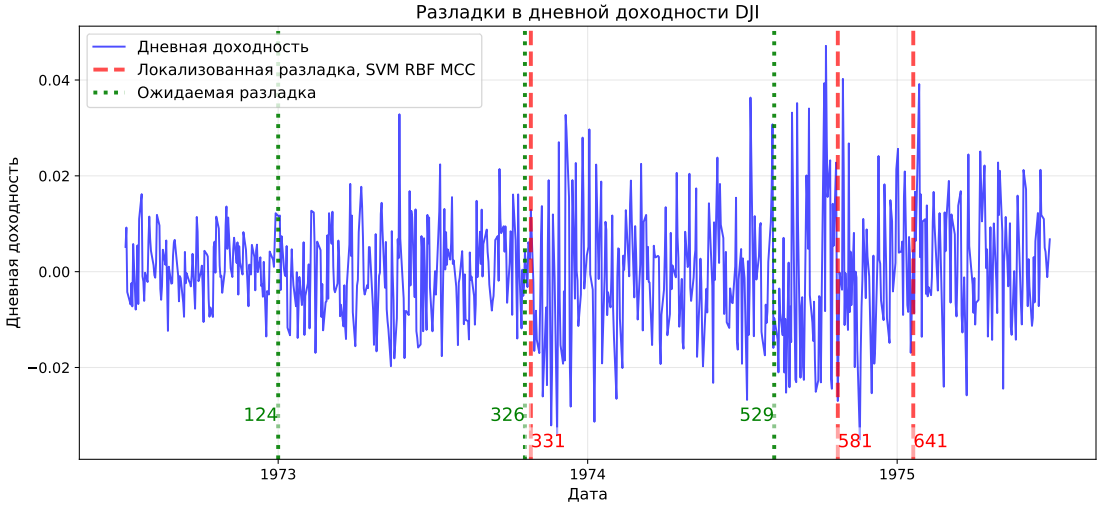
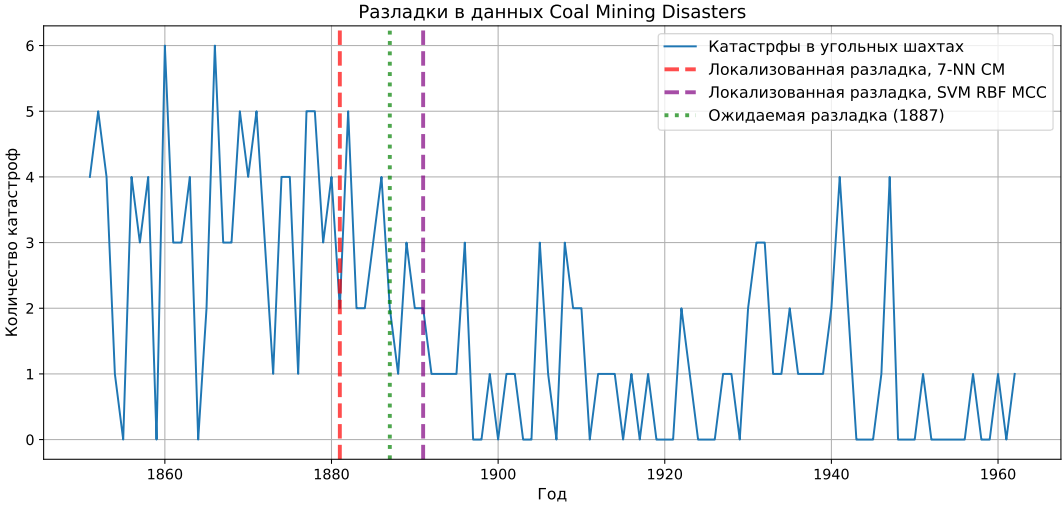
Алгоритм на основе KNN графа: 100% обнаружение, 3 ложных срабатывания



Алгоритм на основе SVM: 100% обнаружение, 1 ложное срабатывание



DJI Returns, алгоритм SVM RBF MCC



Coal Mining Disasters, алгоритмы SVM RBF MCC и 7-NN