Санкт-Петербургский государственный университет
Кафедра системного программирования

Разработка инструмента для автоматической предобработки временных рядов с адаптивными стратегиями

Шангареев Рустам Ильшатович

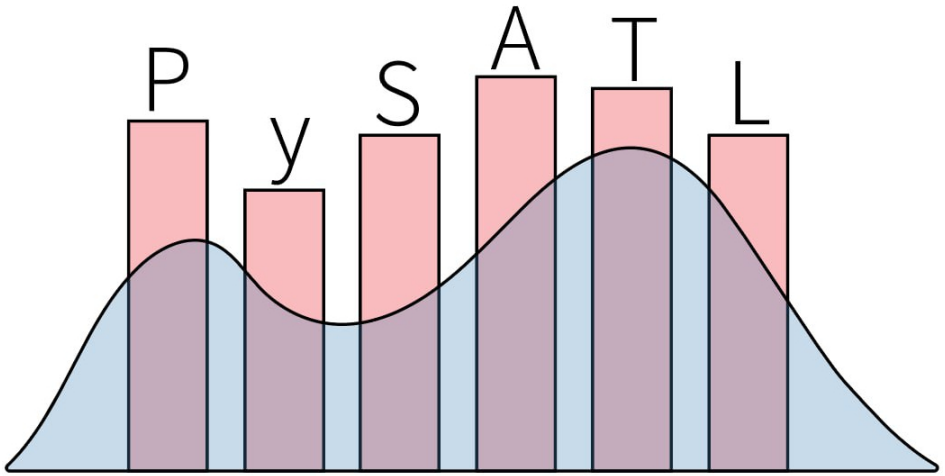
Научный руководитель: ассистент кафедры системного программирования В.И. Гориховский
Консультант: Р.С. Пусев, Инженер ключевых проектов, ООО «МПГ АйТи Солюшнз»

Санкт-Петербург
2025

Введение. PySATL

PySATL — проект по алгоритмической статистике для RnD исследований:

- **Единая архитектура**
- Общий интерфейс по работе с данными
- Конфигурируемое исследование и анализ алгоритмов
- Автоматизация RnD исследований



Цели и задачи

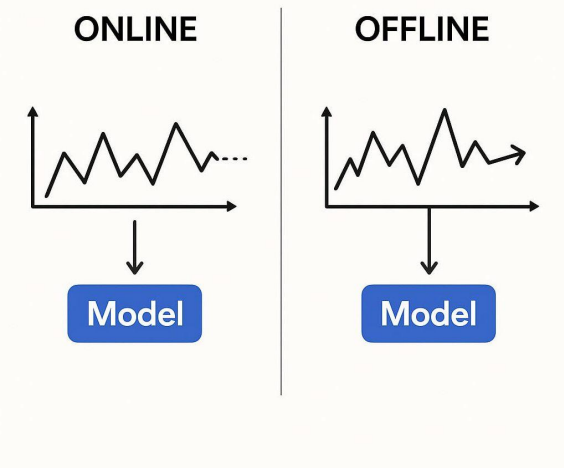
Целью: разработка инструмента для адаптивной предобработки временных рядов для проекта PySATL. Задачи:

1. Провести анализ требований к системе со стороны разработчиков PySATL
2. Провести общий обзор предметной области, существующих решений и используемых в них архитектур обработки временных рядов
3. Разработать архитектуру инструмента для обработки временных рядов согласно функциональным требованиям
4. Реализовать инструмент согласно разработанной архитектуре и требованиям
5. С помощью реализованного инструмента провести интеграционное исследование предобработки на алгоритмах разладки

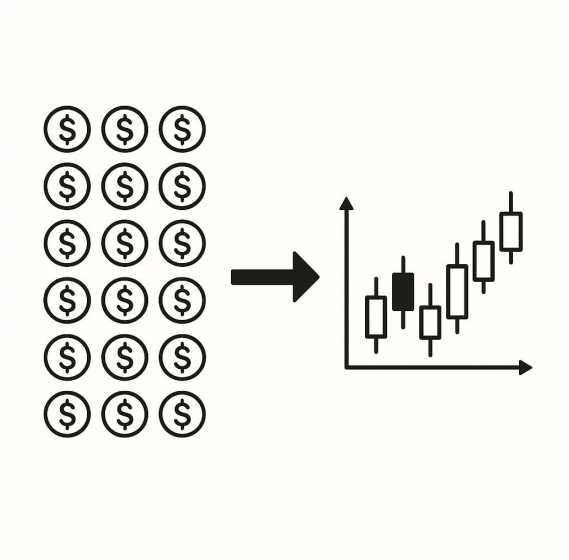
Обзор предметной области

Обработка временных рядов имеет одну из ключевых ролей по следующим причинам:

Различные интерфейсы алгоритмов

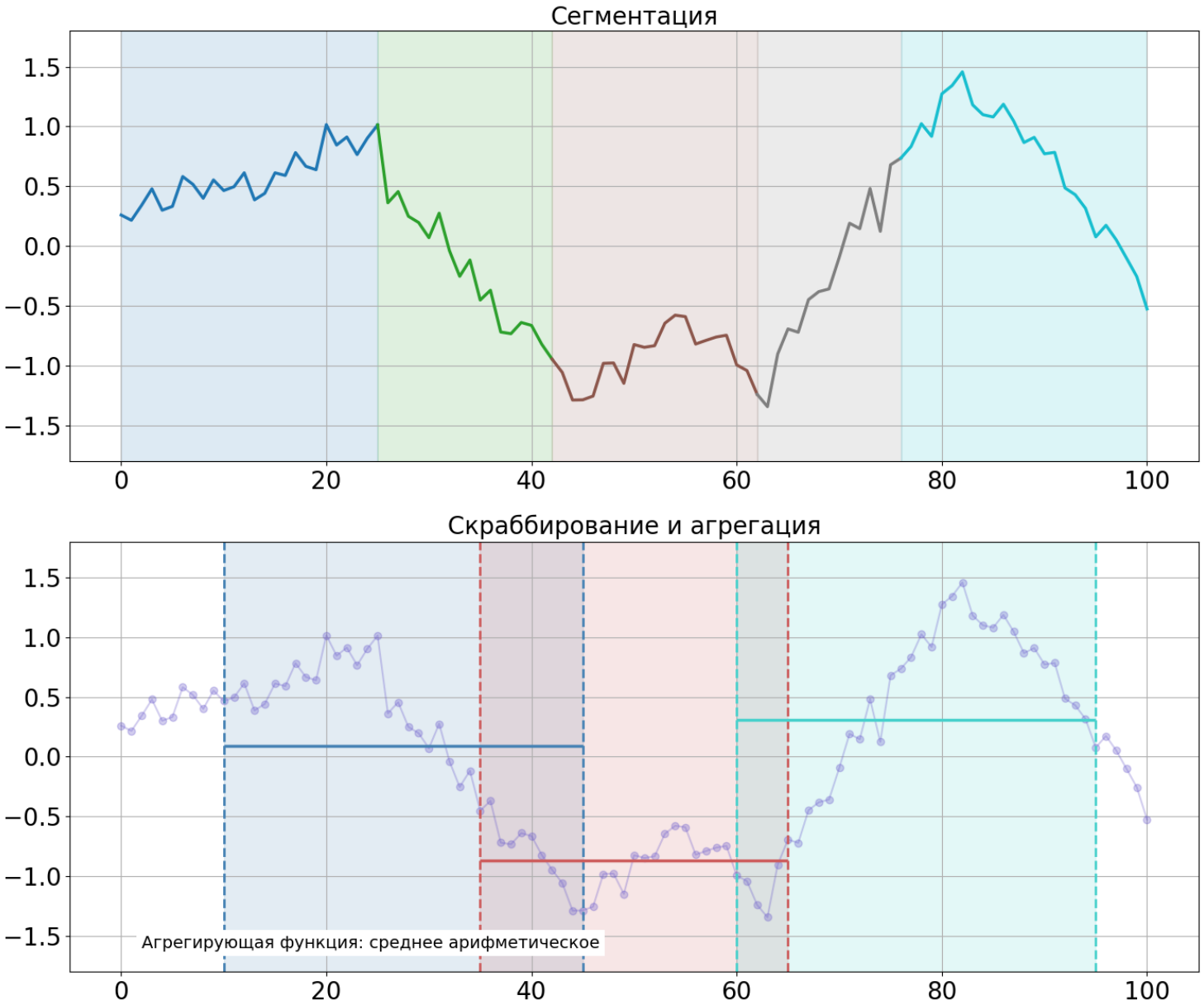


Высокая размерность в данных и шум



Концепт «garbage in, garbage out»

Основные виды предобработки. Иллюстрация



Обзор существующих решений

Проведенное исследование существующих решений отобрало 23 инструмента и разделило их на 4 типа:

1. Инструменты с отсутствием какой-либо функциональности по обработки временных рядов
2. Инструменты ограниченные табличной группировкой
3. Финансовые инструменты с конфигурируемой онлайн и оффлайн обработкой узкой функциональности
4. Код к научным статьям, предлагающий реализацию функций для новых методов предобработки, без какой-либо архитектуры

Требования к библиотеке

- Нефункциональные
  - Ленивая работа с произвольными потоковыми данными
  - Возможности для расширения и совместимость с другими библиотеками
  - Документированность и покрытие модульными тестами в 95 %
- Функциональные
  - Единый интерфейс для компонентов, выполняющих задачу передачи данных на обработку из различных источников: коллекция, файл, WebSocket, база данных
  - Онлайн и оффлайн компоненты для скраббирования, выборки, агрегации, сглаживания и фильтрации
  - Индуктивные и индикаторные обработчики, применяемые в финансовой сфере (19 реализаций)
  - Возможность создания новых обработчиков с помощью конвейерной обработки

Инструментарий

В качестве инструментария были выбраны технологии, используемые в проекте PySATL:

1. Язык программирования: Python
2. Управление зависимостями: Poetry
3. Библиотека для анализа типов: MyPy
4. Библиотеки для тестирования компонентов: Pytest, pytest-mock, hypothesis
5. Библиотеки для оценки покрытия модульным тестированием компонентов: Pytest-cov

Диаграмма компонентов инструмента

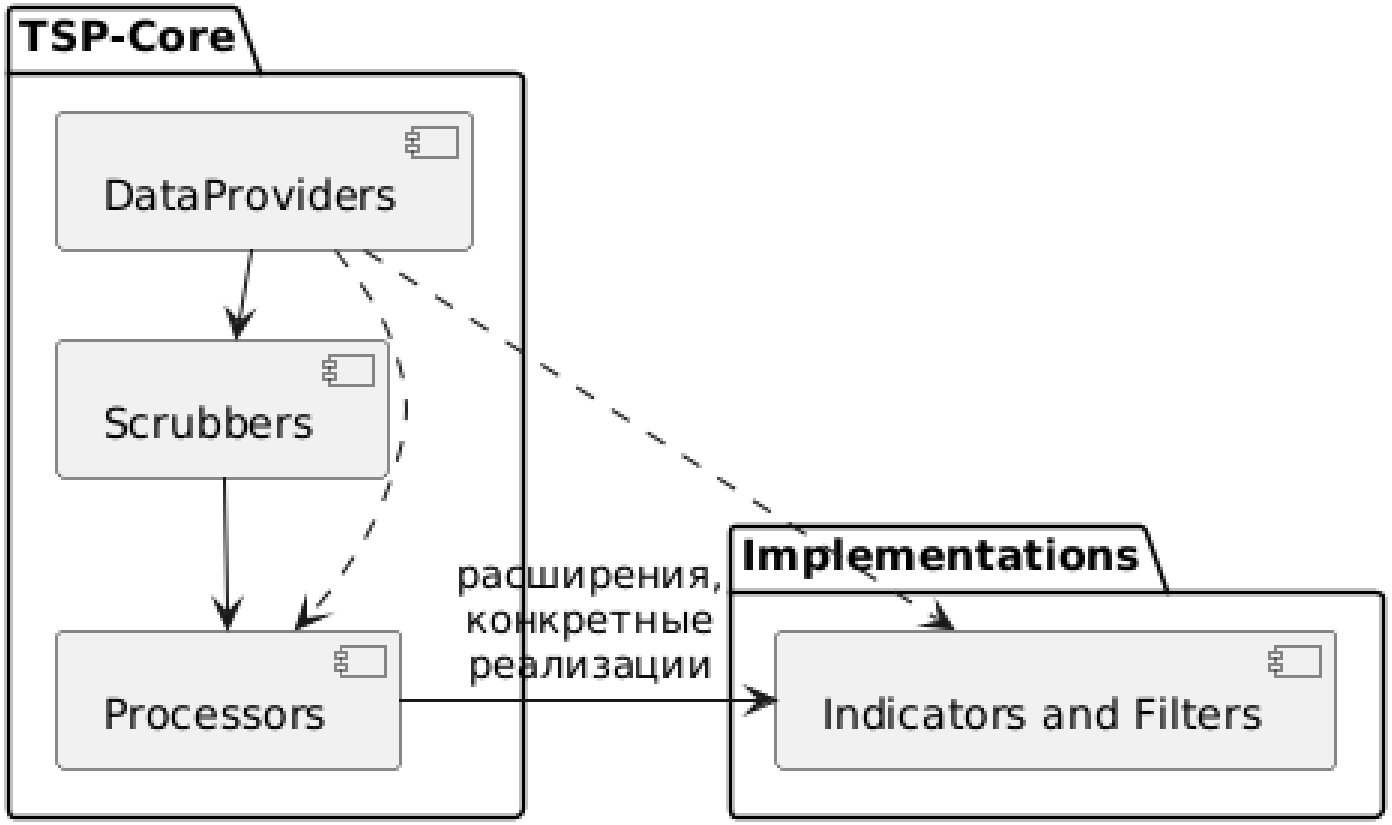
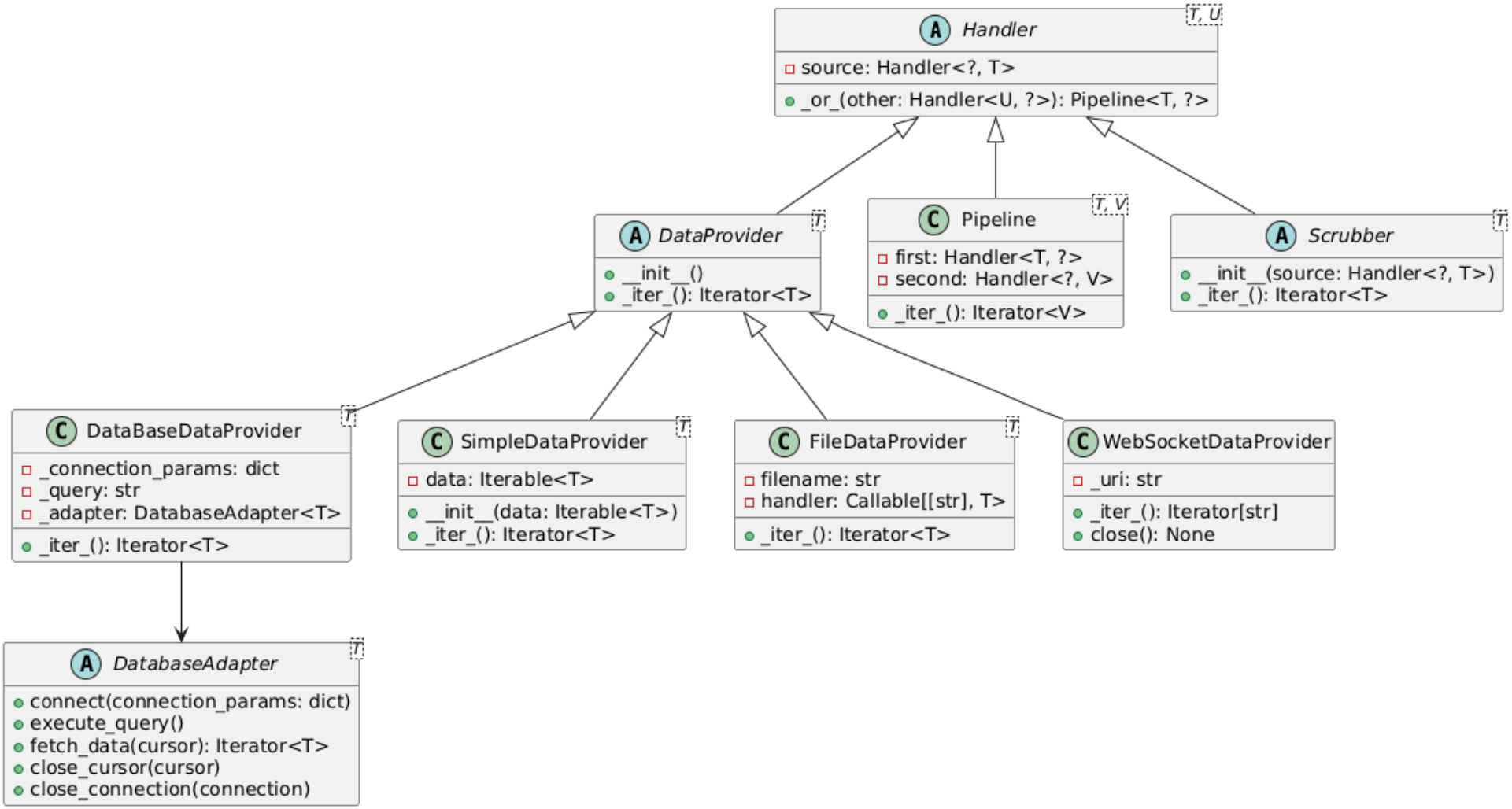


Рис.: «DataProviders» — поставщики данных; «Scrubbers» — скраббирующие обработчики; «Processor» — все остальные обработчики

Диаграмма классов TSP-Core



Содержимое TSP-Core.processors

В TSP-Core.processors находятся

- Комбинирующие обработчики
  - MappingHandler — map обработчик для применения произвольной функции к элементам временного ряда
  - CombineHandler — обработчик для создания преобразователей, комбинирующих результаты обработки несколькими обработчиками
  - TeeHandler — обработчик для создания преобразователей, работающих одновременно с обработанным и необработанным рядами
- Индуктивные и фильтрующие обработчики
  - InductiveFilter — обработчик, реализующий индуктивные функции на временных рядах
  - LagHandler — обработчик для создания преобразователей, работающих с некоторой задержкой «lag»
  - OnlineFilterHandler и OfflineFilterHandler — онлайн и оффлайн, соответственно, фильтрующие обработчики

Интеграционное исследование

Были проведены эксперименты[1] со следующими вопросами:

RQ1: Как меняются метрики качества решения задачи обнаружения разладки у алгоритмов PySATL-CPD после интеграции с библиотекой для предобработки временных рядов?

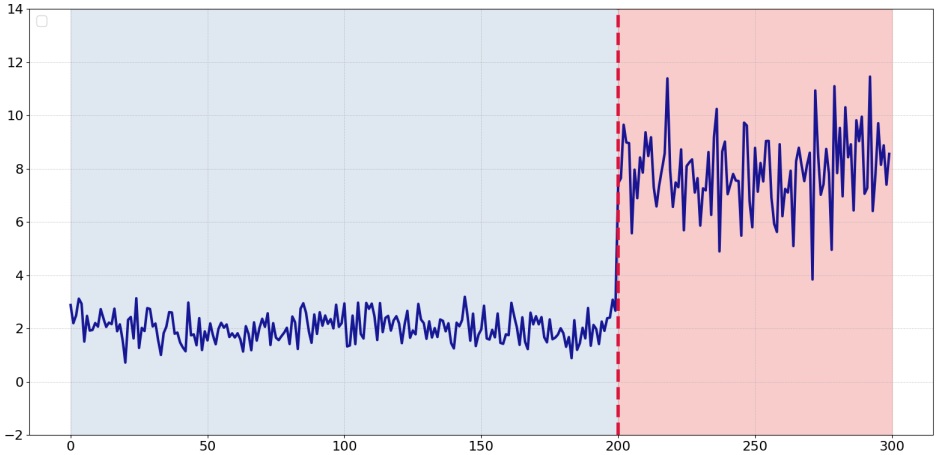
RQ2: Насколько стабильна библиотека для предобработки временных рядов в интеграции с библиотекой для обнаружения разладки?

[1] проведены лично мной из-за высокой загруженности других разработчиков PySATL; апробация с опросом разработчиков в процессе

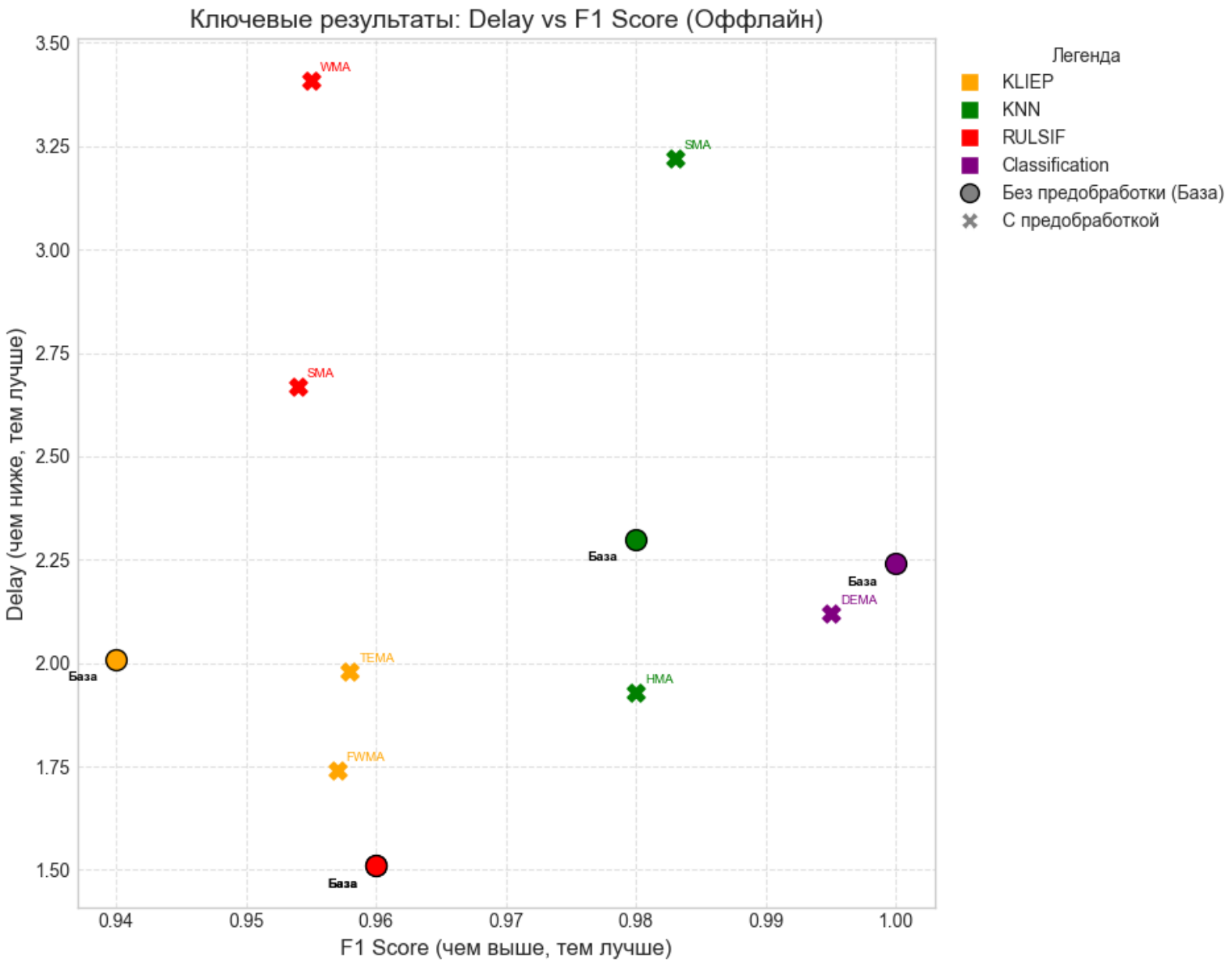
RQ1. Исследование качества

Проведено сравнение качества работы алгоритмов обнаружения разладки на 45 различных наборах синтетических данных по следующим метрикам качества:

- Delay — среднее значение задержки обнаружения разладки
- Detected — процент обнаруженных разладок
- Precision — точность обнаружения разладки (доля истинно-положительных срабатываний среди всех положительных)
- F1 — гармоническое среднее между Precision и Detected
- FPR — доля ложно-положительных срабатываний



RQ1. Результаты исследования качества (оффлайн)



RQ1. Результаты исследования качества (онлайн)

Алгоритм	Delay	Detected	Precision	F1	FPR
Graph	4.17	80.00	0.418	0.512	0.002
KLIEP	2.05	97.78	0.809	0.835	0.000
KNN	5.07	100.00	0.203	0.330	0.009
RULSIF	1.93	97.78	0.622	0.705	0.001
Classification	7.62	88.89	0.371	0.477	0.007

Режим работы: Онлайн. Предобработка: WMA. Различия проанализированы с помощью критериев Уилкоксона и парным t-тестом с уровнем значимости 5%. Зеленый цвет — стат. значимое улучшение; серый цвет — не стат. значимые изменения; красный цвет — стат. значимое ухудшение

RQ2. Исследование стабильности

Алгоритмы запущены в интеграции с PySATL-CPD непрерывно на протяжении 30 минут. Каждые 10000 подряд идущих элементов временного ряда были сняты замеры времени их обработки и потребляемоей памяти с помощью tracemalloc.

Метрики стабильности:

- Отношение дисперсии времени работы к среднему времени работы
- P-value теста Шапиро-Уилка на выборке времени работы
- Максимальное потребление памяти среди всех замеров и p-value теста Манна-Кендалла наличия тренда во временном ряде

RQ2. Результаты исследования стабильности

Результаты экспериментов стабильности:

- Непрерывная работа алгоритмов предобработки в интеграции с PySATL-CPD на протяжении 30 минут
- Дисперсия времени работы у всех алгоритмов составляет не более 8 % от среднего
- Тест Шапиро-Уилка на нормальность распределения: p-value = 0.4
- Среди всех замеров потребление памяти не превышало 120 МБ, а также отсутствует тренд возрастания потребления памяти: p-value = 0.21 в тесте Манна-Кендалла

Заключение

Выполнены следующие задачи:

- Проведен общий обзор предметной области, проекта PySATL, существующих решений и используемых в них архитектур обработки временных рядов
- Разработана архитектура инструмента для обработки временных рядов согласно функциональным и нефункциональным требованиям
- Реализован инструмент согласно разработанной архитектуре и функциональным требованиям
- Проведена апробация инструмента с помощью интеграционного тестирования с PySATL-CPD

Код проекта расположен на GitHub: github.com/pysatl/pysatl-tsp

Диаграмма скраббирующих обработчиков

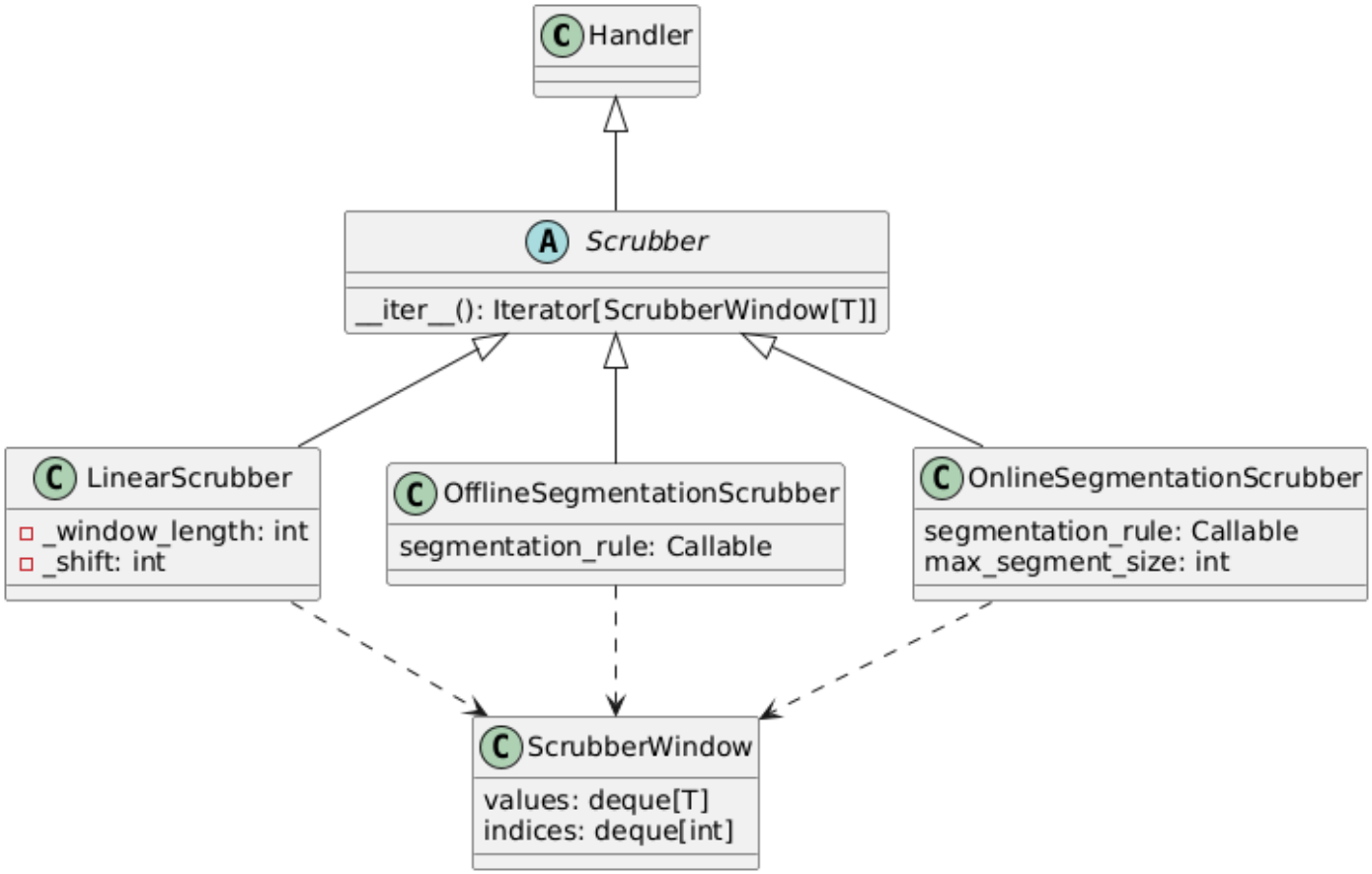
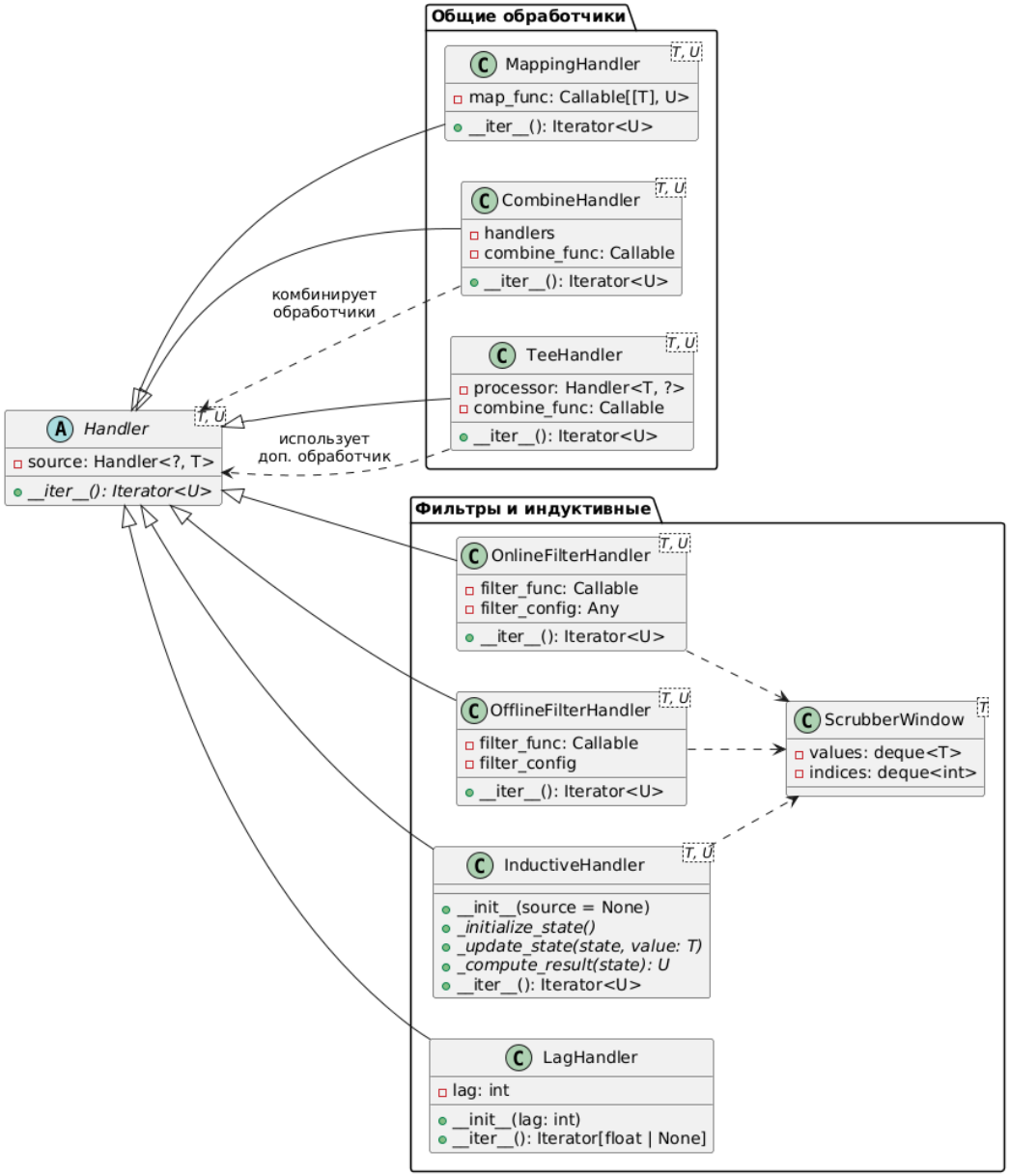


Рис.: «Scrubber» — абстрактный скраббирующий обработчик; «LinearScrubber» — линейный скраббер со скользящим окном; «OnlineSegmentationScrubber» и «OfflineSegmentationScrubber» — онлайн и оффлайн, соответственно, компоненты для сегментации временных рядов; «ScrubberWindow» — специальный data класс для хранения окон из скрабберов

Диаграмма обработчиков



Критерии сравнения существующих решений

Таблица: Легенда кодовых вердиктов

Цвет	Код	Описание
	A: Базовая агрегация	Простая группировка с константными правилами. Это агрегация временных рядов (или финансовых данных) с отбором по константому значению признака.
	B: Код научных статей	Реализация методов предобработки, описанных в научных публикациях
	C: Отсутствие функциональности	Отсутствие встроенной поддержки методов предобработки

Обзор существующих решений

Таблица: Сводная таблица функциональности инструментов 1

Инструмент	Оффлайн	Онлайн
Palantir	A	A
Elastic	A	C
Apache Spark	A	C
PipelineDB	A	C
Wellmann projects	A	A
Ta4J	A	A
PandasTA	A	C
Tsam	A	C
ClasPy	C	C
Skforecast	C	C
TSA in R	A	C
Julia	A	C
TsDownsample	A	C

Обзор существующих решений 2

Таблица: Сводная таблица функциональности инструментов 2

Инструмент	Оффлайн	Онлайн
PATSS	B	C
PyTS	A	C
MatrixProfile	A	C
NeuroKit2	C	C
HeartPy	C	C
TA-Lib	A	C
QuantLib	C	C
Kats	C	C
MetPy	C	C
Flow Forecast	C	C