

Санкт-Петербургский государственный университет

Кафедра системного программирования

Группа 23.Б10-мм

Развитие библиотеки `pysatl-mpest`

Тотьмянин Данил Денисович

Отчёт по учебной практике
в форме «Решение»

Научный руководитель:
доцент кафедры системного программирования, к. ф.-м. н., В. И. Гориховский

Санкт-Петербург
2025

Оглавление

Введение	3
1. Постановка задачи	4
2. Расширение класса распределений	5
2.1. Обзор распределений	5
2.2. Оценка применимости методов	6
3. Пайплайн проведения экспериментов	8
4. Архитектура экспериментального окружения	10
4.1. Этап генерации	10
4.2. Этап оценки	11
4.3. Этап анализа	12
5. Эксперимент	14
5.1. Постановка эксперимента	14
5.2. Результаты	14
6. Требования к архитектуре	16
Заключение	17
Список литературы	18

Введение

В современных прикладных задачах из различных областей активно используются методы математической статистики. Одним из таких методов является оценка параметров смеси распределений [1]. Решение этой задачи требует специализированных инструментов, способных эффективно работать со смесями и обеспечивать точную оценку их параметров. Именно таким инструментом выступает библиотека `pysatl-mpest`, которая является частью крупного и активно развивающегося статистического пакета `PySATL`.

В рамках предыдущей учебной практики в библиотеку был успешно интегрирован ELM (ранее LM-EM) алгоритм, использующий L-моменты [2], [3] для оценки параметров смеси. Был проведен эксперимент, в рамках которого выяснилось, что новый метод значительно быстрее и в некоторых случаях точнее классического EM (далее MLE-EM) алгоритма [4]. Результаты этих исследований были успешно представлены в публикации на 37-й конференции FRUCT [5], что подчёркивает научную значимость и практическую применимость разработанного подхода.

Однако, в процессе дальнейшего развития пакета `PySATL` были выявлены внешние архитектурные ограничения `pysatl-mpest`. Эти ограничения заключаются в невозможности полноценного взаимодействия библиотеки с другими модулями пакета. В частности, выявлена невозможность прямой интеграции в качестве подмодуля в `pysatl-experiment` — ключевой компонент пакета, предназначенный для проведения экспериментов [6]. Кроме того, внутренняя архитектура `pysatl-mpest` демонстрирует недостаточную степень расширяемости, что затрудняет реализацию новых методов.

1. Постановка задачи

Целью работы является развитие функциональности и сбор требований для будущего рефакторинга `pysatl-mpest`. Для её выполнения были поставлены следующие задачи:

- Расширение набора поддерживаемых распределений, а именно распределений с ограниченным носителем или тяжёлыми хвостами.
- Реализация прототипа экспериментального окружения.
- Проведение расширенного эксперимента на стандартных смесях для валидации результатов предыдущей работы.

2. Расширение класса распределений

Для выявления ограничений архитектуры было решено добавить новые типы распределений, каждый со своими особенностями. В этом разделе будет проведён обзор реализованных распределений и проведена оценка применимости методов в контексте текущей архитектуры.

2.1. Обзор распределений

2.1.1. Равномерное распределение

Имеет плотность:

$$f(x \mid a, b) = \frac{1}{b - a}, \quad b > a, \quad x \in [a, b]$$

Классический пример распределения с ограниченным носителем, что позволяет изучить методы на данных, имеющих чёткие границы, в отличие от данных, сгенерированных из распределений с бесконечным носителем.

2.1.2. Несмещённое бета-распределение

Имеет плотность:

$$f(x \mid \alpha, \beta) = \frac{1}{B(\alpha, \beta)} x^{\alpha-1} (1-x)^{\beta-1}, \quad \alpha, \beta > 0, \quad x \in [0, 1]$$

где $B(\alpha, \beta)$ — бета-функция

Является нетривиальным распределением с ограниченным с обеих сторон носителем.

2.1.3. Распределение Коши

Имеет плотность:

$$f(x \mid \gamma, x_0) = \frac{1}{\pi} \cdot \frac{\gamma}{(x - x_0)^2 + \gamma^2}, \quad \gamma > 0, \quad x \in \mathbb{R}$$

Это распределение является примером распределения с тяжёлыми хвостами и имеет колоколообразную форму плотности, схожую с нормальным распределением, что позволяет в дальнейшем изучить возможность разделять смеси распределений Коши и нормального.

2.1.4. Распределение Парето I типа

Имеет плотность:

$$f(x \mid k, x_m) = \frac{kx_m^k}{x^{k+1}}, \quad k > 0, \quad x \geq x_m$$

Как и распределение Коши, распределение Парето I типа характеризуется тяжёлыми хвостами, хотя и только при $k < 1$. Его включение позволит оценить применимость и точность алгоритмов на данных с выраженной асимметрией и потенциальными выбросами.

2.2. Оценка применимости методов

На таблице 1 приведены результаты оценки. Архитектурная структура алгоритмов без дополнительных изменений применима только для распределений Коши и несмещённого Бета. Для оценки параметров равномерного распределения и Парето I типа необходима реализация специализированных типов алгоритмов.

Распределение	MLE-EM	ELM
Коши		
Равномерное		
Несмещённое бета		
Парето I типа		

Таблица 1: Результаты анализа применимости различных распределений в контексте архитектуры (MLE-EM и ELM)

Помимо этого, выявлена проблема несовместимости компонент в контексте алгоритмов: если смесь состоит, например, из распределения Коши и нормального распределения, то ELM алгоритм невозможно

применить к ней ввиду его теоретической неприменимости к распределению Коши.

Одной из ключевых задач при оценке параметров распределений является их корректная обработка, особенно при использовании оптимизаторов, требующих нормализации входных данных. Тогда как для большинства распределений она требуется, существуют распределения, для которых нет необходимости в нормализации. Это расхождение не имеет универсального решения в данной архитектуре, что является проблемой.

3. Пайплайн проведения экспериментов

В этом разделе будет описан процесс проведения эксперимента, структурированный в последовательный пайплайн, который частично воспроизводит структуру эксперимента из `pysatl-experiment`.

- **Этап 0: Конфигурирование**

Предназначен для фиксирования общих параметров эксперимента, к которым относятся: размер выборки, семейства распределений, количество повторных запусков (итераций) для каждой конфигурации и выбор начального сйда для всех случайных процессов.

- **Этап 1: Генерация данных**

Цель данного этапа — создание синтетических выборок из смесей распределений. Для каждого из запусков (определенных в этапе 0) и каждой конфигурации из этапа 0 осуществляется сэмплирование данных, включающее:

- Генерацию случайных весов Ω и параметров Θ для компонент смеси.
- Сэмплирование из сгенерированной смеси.
- Сохранение результатов для дальнейшей оценки.

- **Этап 2: Оценка параметров**

На этом этапе применяются алгоритмы к выборкам из предыдущего этапа. Для каждой выборки генерируется случайный набор начальных весов и параметров смеси. Важно отметить, что в случае сравнения алгоритмов необходимо установить одинаковый сид, чтобы сравнение было корректным. Далее запускаются алгоритмы для оценки параметров смеси и сохраняются результаты их работы.

- **Этап 3: Анализ результатов**

Для результатов из второго этапа возможен анализ в режиме сравнения двух алгоритмов или в режиме оценки конкретного алгоритма. Анализ включает вычисление метрик качества и времени выполнения алгоритмов. Из этих значений формируются выборки, для которых затем вычисляются описательные статистики: среднее, медиана, стандартное отклонение и процентиля.

Все промежуточные и финальные данные на каждом этапе сохраняются в файлах форматов .csv и .yaml для обеспечения возможности дальнейшего анализа и полной воспроизводимости. Таким образом, при наличии данных с первого или второго этапа нет необходимости проводить их повторно.

4. Архитектура экспериментального окружения

Описанный в секции 3 пайплайн реализован в виде модульной архитектуры, которая будет описана в текущем разделе.

Для воспроизводимости экспериментов необходимо зафиксировать различные сиды для первого и второго этапа, передав их в конструкторы соответствующих классов. Высокоуровневая архитектура экспериментального окружения представлена на Рис. 4.

4.1. Этап генерации

Эта часть экспериментального окружения отвечает за сэмплирование датасетов из случайно сгенерированных смесей.

Прежде всего генерируется смесь распределений при помощи объекта класса `AMixtureGenerator`. Конкретными реализациями являются:

- **DatasetMixtureGenerator**: Генерирует для каждой компоненты смеси параметры из равномерного распределения. Веса компонент генерируются из распределения Дирихле с параметрами, при которых значения, близкие к граничным, генерируются намного реже. Такой подход необходим для минимизации сценариев, в которых компоненты смеси вырождаются в ходе работы алгоритма.
- **RandomMixtureGenerator**: То же, что и `DatasetMixtureGenerator`, но веса каждой компоненты равны.
- **StandardMixtureGenerator**: Генерирует смесь со стандартными параметрами (например, нормальное распределение с параметрами $\mu = 0$, $\sigma = 1$).

Далее пользователь конфигурирует генератор датасетов, фиксирует смеси, для которых эксперимент будет проводиться, количество запусков и размер выборки.

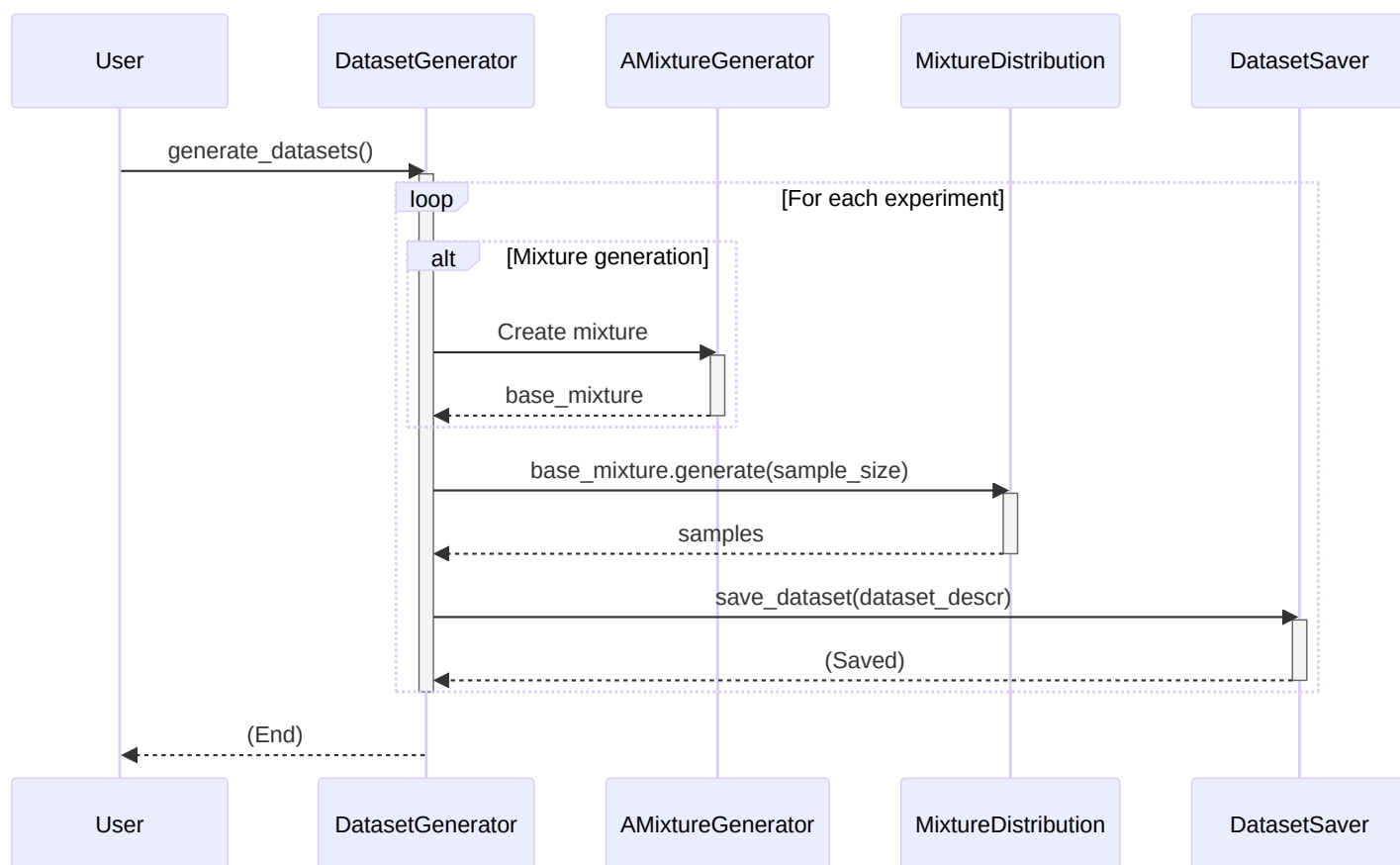


Рис. 1: Диаграмма потоков данных этапа генерации

DatasetGenerator сэмплирует выборки в соответствии с конфигурацией, после чего объект класса DatasetSaver сохраняет результаты в указанную директорию.

4.2. Этап оценки

Данный этап предназначен для оценки параметров смеси и сохранения результатов. Класс DatasetParser загружает данные с первого этапа эксперимента, которые передаются в AExecutor для запуска алгоритма.

Алгоритмы запускаются внутри объекта класса AEstimator. Для ускорения вычислений было решено использовать ProcessPoolExecutor из пакета multiprocessing.

Результаты работы алгоритма сохраняются при помощи объекта

класса ExperimentSaver.

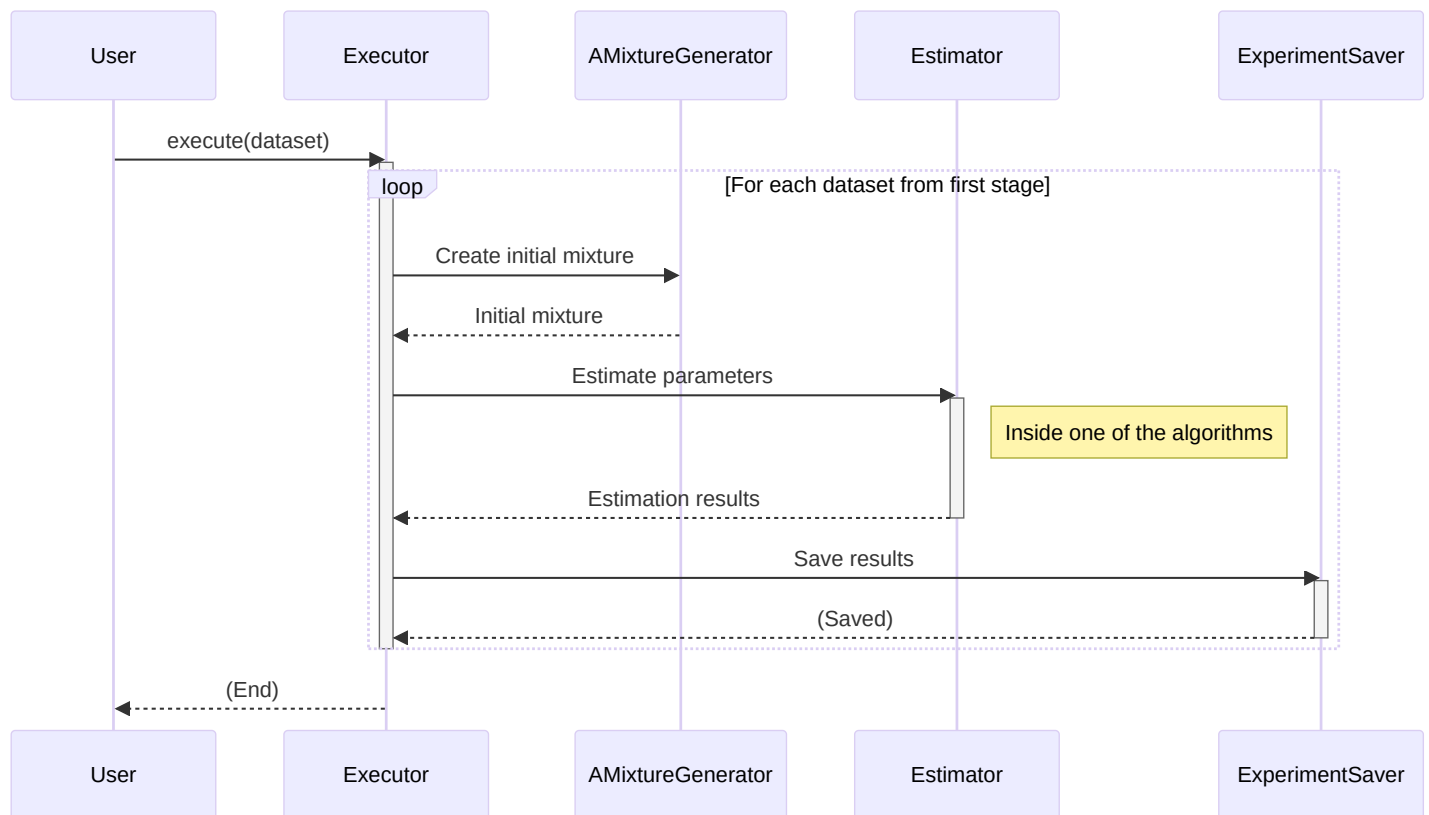


Рис. 2: Диаграмма потоков данных этапа оценки

4.3. Этап анализа

Анализ возможен в двух режимах: *analyze* и *compare*. Первый предназначен для получения информации о работе одного алгоритма, тогда как второй предназначен для их сравнения.

Пользователь конфигурирует класс `Analysis`, фиксируя методы анализа двух видов:

- **AnalysisStrategy**: Предназначен для анализа единичного запуска. Конкретными реализациями являются:
 - **DensityPlot**: Сохраняет график истинной и результирующей смеси, а также гистограмму выборки.

- **ErrorConvergence**: Сохраняет график зависимости значения метрики от шага алгоритма. В качестве метрики сейчас доступен только Mean integrated squared error (MISE).
- **TimePlot**: Сохраняет график времени выполнения каждого шага алгоритма.
- **AnalysisSummarizer**: Предназначен для анализа всех запусков конкретной смеси. Включает в себя:
 - **ErrorSummarizer**: Сохраняет статистики по выборке метрик, такие как среднее, стандартное отклонение, медиану и 90, 95% процентиля.
 - **TimeSummarizer**: Сохраняет среднее время выполнения и стандартное отклонение.

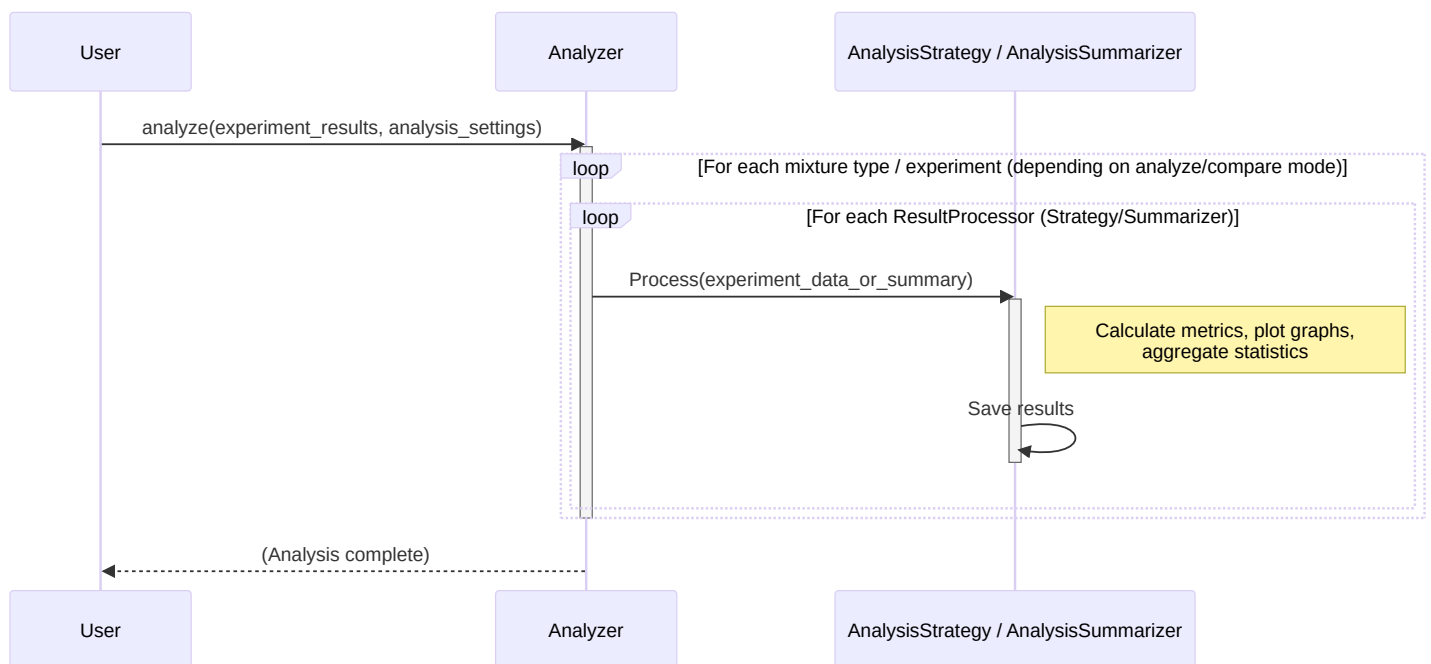


Рис. 3: Диаграмма потоков данных этапа анализа

5. Эксперимент

5.1. Постановка эксперимента

Для подтверждения воспроизводимости результатов, полученных в рамках работы над прошлой практикой, было решено провести расширенный эксперимент с большим количеством запусков:

- Компоненты, используемые в смесях — экспоненциальное, нормальное, Вейбулла с двумя параметрами.
- Алгоритмы — MLE-EM, ELM.
- Используются все реализованные в пакете `mpest` оптимизаторы для MLE-EM, из результатов которых выбирается смесь с наибольшим сходством с базовой.
- Количество компонент в смесях — 2.
- Количество экспериментов для каждой смеси — 10000.
- Количество шагов алгоритма — 16.
- Количество элементов в выборках - 200, 1000.

Данные сгенерированы с фиксированным сидом 42, оценка параметров производилась с фиксированным сидом 43.

В качестве метрики для эксперимента используется MISE:

$$\int_{-\infty}^{+\infty} (p(x | \Theta) - p(x | \hat{\Theta}))^2 dx$$

где $p(x | \Theta)$ — плотность базовой смеси с параметрами Θ , $p(x | \hat{\Theta})$ — плотность смеси с оценёнными параметрами $\hat{\Theta}$.

5.2. Результаты

Результаты эксперимента приведены в таблице 2.

Размер	Смесь	ELM					MLE-EM				
		μ	σ	50%	90%	95%	μ	σ	50%	90%	95%
200	Exp+Exp	0.006	0.009	0.0027	0.015	0.022	0.006	0.009	0.0027	0.015	0.022
	Exp+Norm	0.01	0.18	0.0028	0.015	0.024	0.066	0.13	0.016	0.18	0.32
	Norm+Norm	0.0068	0.042	0.0014	0.009	0.021	0.01	0.05	0.002	0.012	0.031
	Weib+Norm	3.6	94	0.003	0.052	0.22	5	257	0.023	0.34	0.64
	Weib+Weib	12	858	0.008	0.59	1.4	3.1	160	0.007	0.33	0.88
1000	Exp+Exp	0.001	0.002	0.0007	0.004	0.005	0.001	0.002	0.0007	0.004	0.005
	Exp+Norm	0.0027	0.021	0.0006	0.003	0.005	0.069	0.14	0.015	0.2	0.37
	Norm+Norm	0.0042	0.028	0.0004	0.0032	0.01	0.0095	0.059	0.0004	0.0062	0.026
	Weib+Norm	130	12500	0.0007	0.025	0.11	2.2	54	0.024	0.35	0.65
	Weib+Weib	6.1	258	0.0021	0.46	1.2	5.9	259	0.0019	0.33	0.85

Таблица 2: Сравнение метрик ошибки ELM и MLE-EM алгоритмов для различных смесей распределений при размере выборки 200, включая среднее, стандартное отклонение, медиана, 90-й и 95-й процентиля.

По данным с таблицы видно, что ELM работает стабильно точнее на размерах выборки 200 и 1000, при этом улучшая свою точность с ростом размера выборки.

Исключением является смесь Вейбулловских распределений, для которых MLE-EM показывает лучшие результаты.

6. Требования к архитектуре

- Гибкость и расширяемость:
 - Универсальность модели для абстракций распределений.
 - Поддержка специализированных версий алгоритмов (реализация для уже имплементированных EM/ELM, поддержка для будущих).
 - Поддержка каскадного использования алгоритмов, то есть использование конкретного алгоритма на конкретной компоненте.
- Надёжность и корректность:
 - Проверка валидности конфигурации алгоритма для конкретной смеси распределений. В частности, проверка на применимость алгоритма к каждой компоненте смеси.
- Совместимость с другими модулями PySATL:
 - Соответствие интерфейсам pysatl-experiment и pysatl-tsp.
 - Интеграция в pysatl-experiment (реализация сценариев генерации, обработки и создания отчётов).

Заключение

- Имплементированы новые семейства распределений, такие как¹:
 - Равномерное распределение
 - Бета-распределение
 - Распределение Коши
 - Распределение Парето I типа
- Был спроектирован и реализован прототип экспериментального окружения, повторяющий структуру эксперимента фреймворка `pysatl-mpest`. Данный прототип позволяет в декларативном стиле проводить эксперименты для анализа и/или сравнения алгоритмов².
- С помощью этой системы был проведен расширенный эксперимент, подтвердивший воспроизводимость полученных в прошлом семестре результатов.
- Сформулирован детальный список требований к будущему рефакторингу архитектуры `pysatl-mpest`.

¹<https://github.com/PySATL/pysatl-mpest/pull/16>

²<https://github.com/PySATL/pysatl-mpest/pull/10>

Список литературы

- [1] Bahram Haji Joudaki, Reza Hashemi, and Soleiman Khazaei. Survival analysis using dirichlet process mixture model with three-parameter burr xii distribution as kernel. *Communications in Statistics - Simulation and Computation*, 53:1–19, 06 2022.
- [2] J. R. M. Hosking. L-moments: Analysis and estimation of distributions using linear combinations of order statistics. *Journal of the Royal Statistical Society. Series B (Methodological)* , 1990, Vol. 52, No. 1 (1990), pp. 105-124, 1989.
- [3] T. Ulrych, Danilo Velis, AD Woodbury, and Mauricio Sacchi. L-moments and c-moments. *Stochastic Environmental Research and Risk Assessment*, 14:50–68, 01 2000.
- [4] Totmyanin Danil. Добавление метода l-моментов в библиотеку mpest. 2025.
- [5] Danil Totmyanin and Viacheslav Gorikhovskii. L-moments method: Introducing novel approach of mixture parameters estimation and its implementation in python library pysatl-mpest. In *2025 37th Conference of Open Innovations Association (FRUCT)*, pages 317–322, 2025.
- [6] Alexey Mironov, Lev Golofastov, and Viacheslav Gorikhovskii. The pysatl experiment framework. In *2025 37th Conference of Open Innovations Association (FRUCT)*, pages 195–202, 2025.

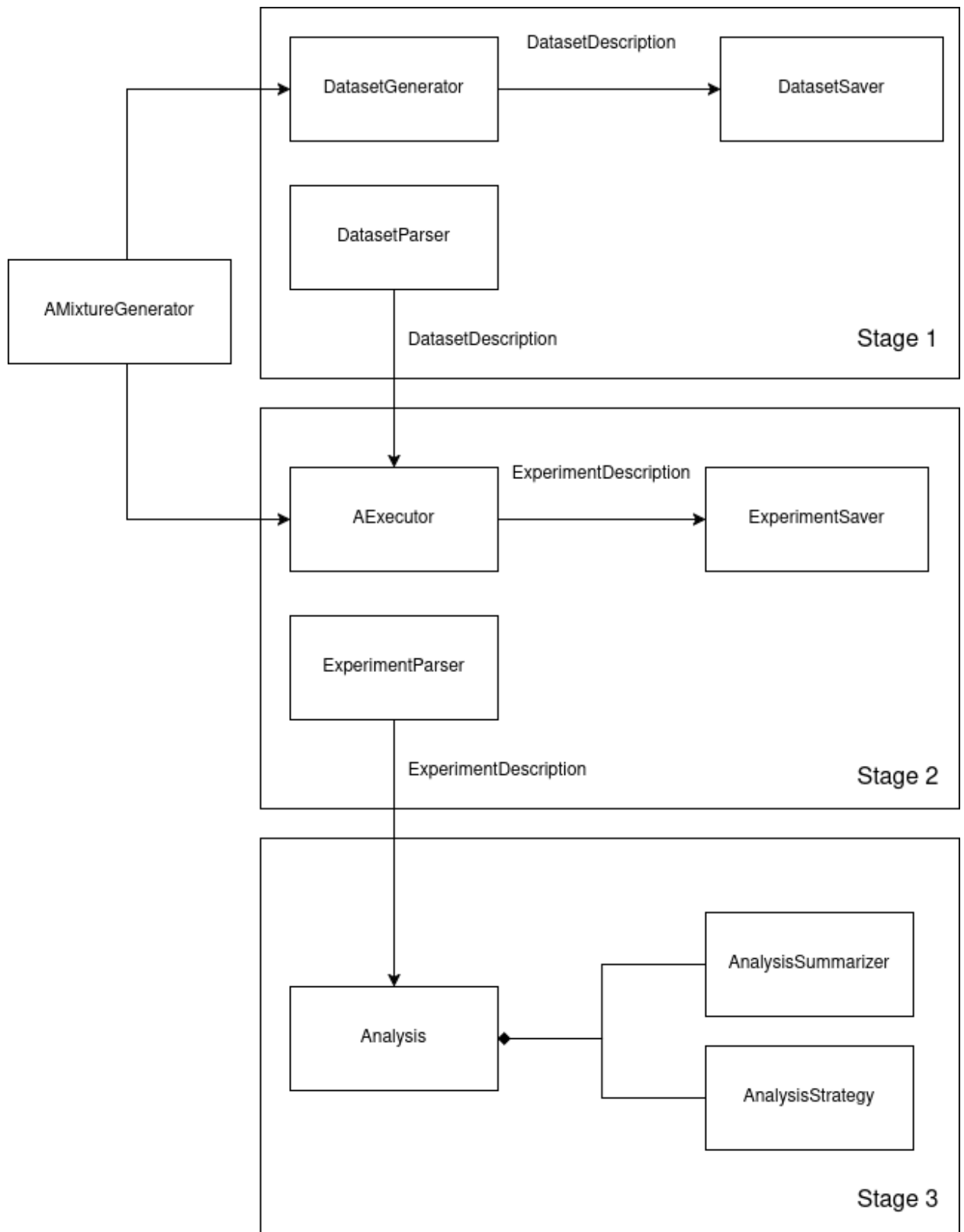


Рис. 4: Высокоуровневая диаграмма архитектуры экспериментального окружения.