

Raport z prac z przedmiotu PDI w semestrze 6

Piotr Jabłoński 325163

25 czerwca 2025

1 Temat pracy

"Biblioteka algorytmów grupowania dla RUST". Implementacja w formacie typowym dla projektów RUST, mianowicie w postaci "skrzyni" **crate**. Repozytorium: <https://github.com/Depermitto/klaster>. Najnowsza wersja możliwa do pobrania: <https://crates.io/crates/klaster>. Praca jest objęta licencją MIT.

2 Wybrane algorytmy do implementacji

2.1 KMeans

- Python: [scikit-learn](#) v1.6.1
- Rust: [linfa-clustering](#) v0.7.1

Najpopularniejszy i potencjalnie najszybszy algorytm klastrowania, szczególnie efektywny przy prostych danych (tabularycznych).

2.2 HDBSCAN

- Python: [scikit-learn-contrib](#) v0.8.40
- Rust: [hdbscan](#) v0.10.1

Dokładniejszy, odporny na różne gęstości danych oraz niewymagający parametru **eps** w porównaniu do najpopularniejszego algorytmu gęstościowego, czyli DBSCAN.

2.3 N2D

- Python: [N2D](#) commit f55316f

Nowe podejście do klastrowania, cechujące się dużą skutecznością na złożonych strukturach danych (obrazy, tekst).

3 Wybrane zbiory danych

1. Syntetyczne: **blobs**
2. Tabularyczne: **BCW**, **winequality-red**
3. Obrazowe: **MNIST**
4. Tekstowe: **20 Newsgroups**

4 Wybrane metryki

1. **ACC** – dokładność przypisania punktów do prawdziwych klas
2. **NMI/AMI** – wzajemna informacja między klastrami a prawdziwymi klasami. AMI koryguje NMI o przypadek losowy
3. **ARI** – skorygowany współczynnik Rand, uwzględniający przypadek losowy

5 Postęp prac

- Przegląd literatury na temat algorytmów klastrujących (KMeans, HDBSCAN oraz wielu podejść z użyciem głębokiego uczenia), zbiorów danych nadających się do uczenia i testowania oraz metryk służących do oceny jakości implementacji tych algorytmów.
- Zapoznanie się z istniejącymi rozwiązaniami dotyczącymi klastrowania w językach Python oraz Rust, wybór tych najbardziej popularnych oraz najlepiej zaimplementowanych.
- Przetestowanie istniejących bibliotek na wybranych zbiorach danych i ich możliwościach, posługując się wybranymi metrykami.
- Zrozumienie i stworzenie **własnej implementacji** algorytmu KMeans. Pierwsza wersja (**v0.0.1-alpha1**) umożliwi inicjalizację centroidów metodami **Forgy** lub **KMeans++** oraz wykorzystanie różnych miar do obliczania dystansu między obserwacjami a klastrami.