scikit-fingerprints: biblioteka w języku Python do efektywnego obliczania fingerprintów molekularnych

Jakub Adamczyk, Piotr Ludynia
Wydział Informatyki, Akademia Górniczo-Hutnicza w Krakowie

Chemoinformatyka

Dziedzina na styku chemii molekularnej i informatyki, dotycząca komputerowego przetwarzania molekularnych. Z perspektywy obliczeniowej są to grafy, z atrybutowanymi wierzchołkami oraz krawędziami.

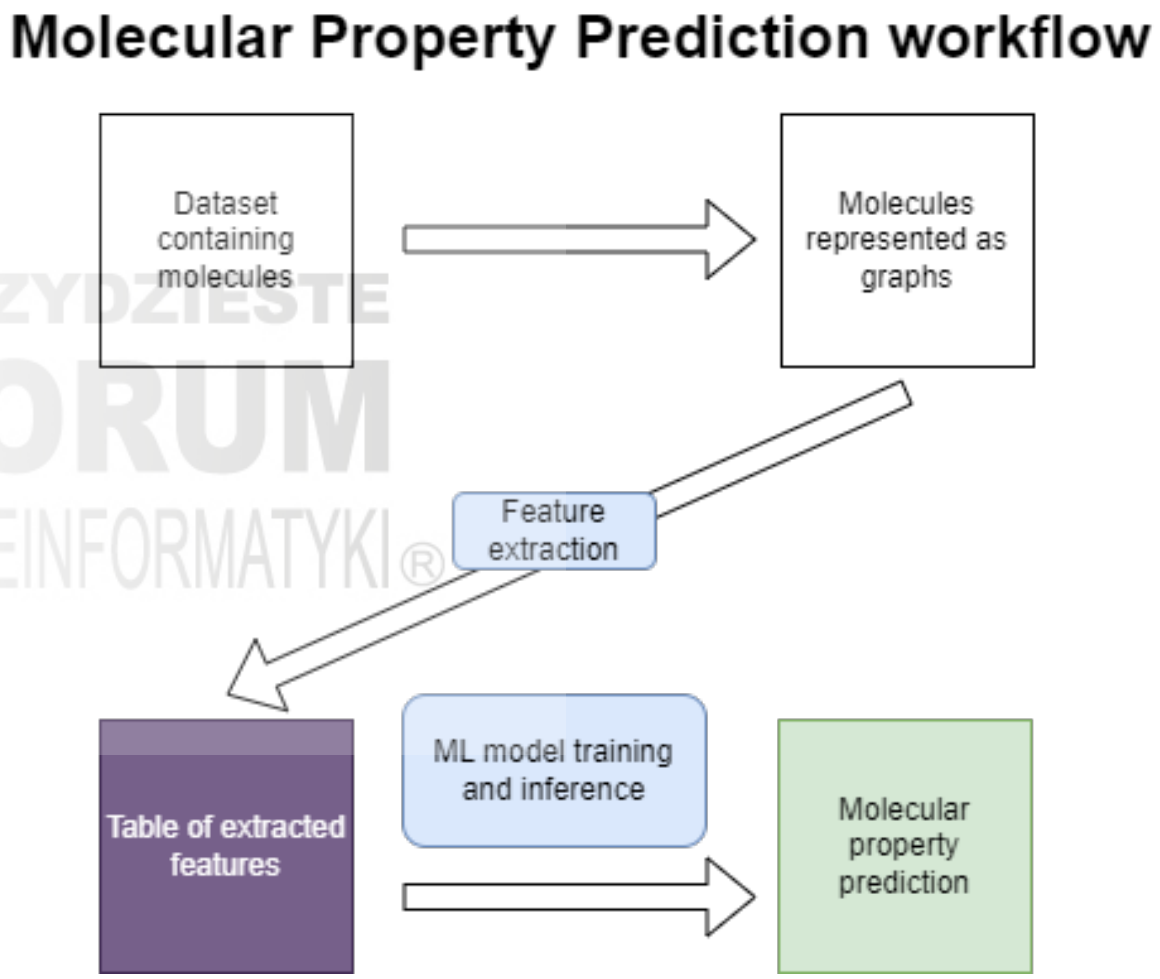
Zajmuje się między innymi ich klasyfikacją, projektowaniem, przewidywaniem własności, analizą danych molekularnych zarówno pojedynczych cząsteczek, jak i całych ich zbiorów. Typowo opiera się przy tym o uczenie maszynowe (ML) oraz sztuczną inteligencję.

Najpowszechniejsze zadania, o ogromnym znaczeniu praktycznym w projektowaniu leków, to wyszukiwanie podobnych molekuł (*virtual screening*) czy przewidywanie własności (*molecular property prediction*).

Projektowanie leków

Metody obliczeniowe są niezbędne w nowoczesnym projektowaniu nowych leków. Ścieżka od początku badań do wdrożenia rynkowego średnio zajmuje ponad 10 lat i kosztuje ok. miliard dolarów. Choćby pandemia COVID-19 pokazała, jak ważne jest przyspieszanie tych procesów.

Sztuczna inteligencja jest wykorzystywana, aby już na wczesnych etapach wykrywać i odrzucać mało obiecujące cząsteczki. Przykładowo, leki muszą być rozpuszczalne w wodzie (predykcja logP), oraz mało toksyczne (klasyfikacja binarna). Można także odkrywać nowe, nieoczywiste dla chemików, poprzez generatywne AI zastosowane do tworzenia grafów o zadanych własnościach.



Przetwarzanie molekuł

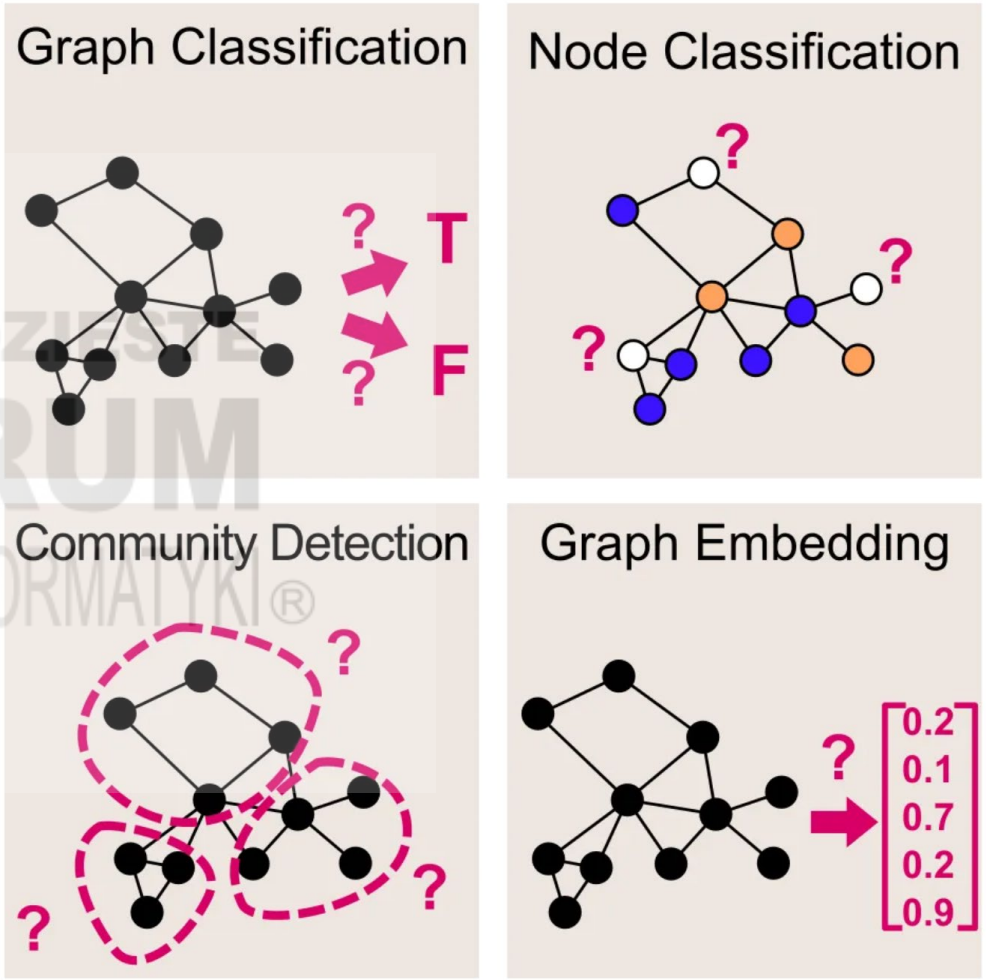
Molekuły są typowo przesyłane w formacie tekstowym SMILES. Z tej reprezentacji można je przekonwertować na grafy, które są zwykle najlepszą chemicznie reprezentacją molekuły. Z perspektywy uczenia maszynowego są jednak dość problematyczne.

Grafy są niezmiennicze ze względu na permutacje wierzchołków i krawędzi. Są także nieeuklidesowe - nie posiadają naturalnej metryki odległości. Klasyfikacja w ML wymaga jednak danych reprezentowanych jako wektory.

Uczenie maszynowe na grafach

Dziedzina ML dotycząca przetwarzania grafów, np. ich klasyfikacji czy regresji, a w szczególności reprezentacji ich za pomocą wektorów (*graph representation learning*). Kluczowe jest, jak przekształcić graf do takiego wektora. Jest tutaj szereg możliwości, np. kernele grafowe (*graph kernels*), grafowe sieci neuronowe (*Graph Neural Networks*, *GNNs*), oraz **fingerprinty molekularne** (*molecular fingerprints*).

W przypadku chemoinformatyki ostatnie lata pokazały, że kernele grafowe mają zwykle zbyt duży koszt obliczeniowy, a sieci GNN są niestabilne i trudne w treningu. Fingerprinty molekularne były oraz pozostają najpopularniejszym podejściem.



Źródło: https://towardsdatascience.com/graph-convolutional-networks-deep-99d7fee5706f

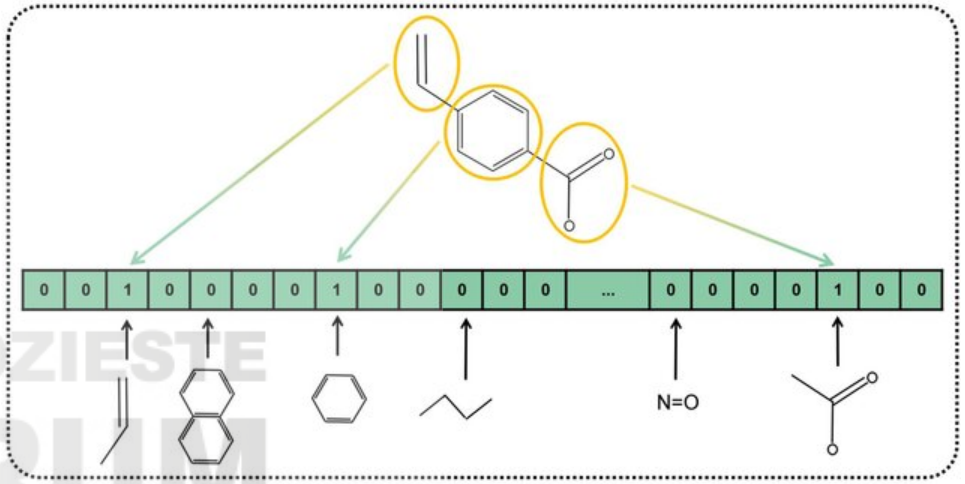
Fingerprinty molekularne

Algorytmy wektoryzacji grafów. Wykorzystują szereg deskryptorów grafowych oraz technik przetwarzania danych, jak np. najkrótsze ścieżki, haszowanie, czy histogramy cech. Wyjściem takich algorytmów są typowo wysokowymiarowe wektory, dobrze reprezentujące własności strukturalne oraz funkcjonalne molekuł. Przekształcenia takie nie są odwracalne i nie zawierają całej wiedzy o grafie, ale często dające wręcz zaskakująco dobre wyniki. Fingerprinty można podzielić w przybliżeniu na dwa rodzaje:

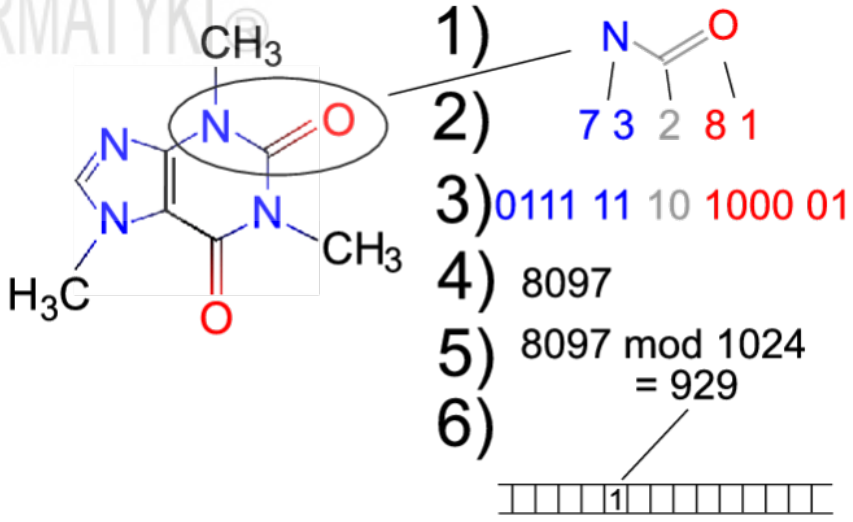
Fingerprinty deskryptorowe (*descriptors / substructural*) - wydobywają stałą liczbę z góry określonych cech. Mogą to być np. liczba atomów ciężkich (poza wodorem) lub wiązań, czy poszczególne grupy funkcyjne lub subgrafy. Wybór takich substruktur jest dokonywany przez ekspertów w dziedzinie. Przykład takiego fingerprintu to MACCS Keys - zestaw 166 cech wybranych dla chemii medycznej.

Fingerprinty haszowane (*hashed*) - definiowany jest wzorzec subgrafów, np. koncentryczne sąsiedztwa, najkrótsze ścieżki, lub drogi. Dla molekuły są obliczane wszystkie wystąpienia takich subgrafów i haszowane na wektor danej długości, zliczając wystąpienia. W celu minimalizacji kolizji typowo używa się długich wektorów, np. 2048. Przykładowe fingerprinty to ECFP (używający koncentrycznych sąsiedztw), Atom Pair (używający najkrótszych ścieżek między wszystkimi parami atomów).

Fingerprint MACCS:



Fingerprint Atom Pair:



Źródła obrazów: http://dx.doi.org/10.3389/fgene.2022.919264, https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-017-1921-4

Problemy dotychczasowych rozwiązań

Fingerprinty molekularne mają ogromną istotność naukową i praktyczną. Niestety, dotychczasowe oprogramowanie do ich obliczania ma wiele istotnych problemów i ograniczeń. Główne z nich to:

1. **Brak równoległości** - każdy współczesny procesor jest wielordzeniowy, a obecne biblioteki obliczają fingerprinty sekwencyjnie. Przy przetwarzaniu molekuł można je jednak przetwarzać bez jakiejkolwiek synchronizacji, a więc problem jest idealnie współbieżnialny. Sekwencyjność obliczeń ogromnie wydłuża czas obliczeń i utrudnia przeprowadzanie wielu badań, np. tuning hiperparametrów w ML.
2. **Brak wsparcia dla macierzy rzadkich** - fingerprinty często skutkują ogromnymi, a przy tym rzadkimi macierzami cech, szczególnie dla długich fingerprintów i większych zbiorów danych. Jest to idealne zastosowanie dla formatów macierzy rzadkich, np. CSR, które obniżają koszt obliczeniowy i pamięciowy. Istniejące oprogramowanie często wspiera jednak tylko macierze gęste, przechowując bez potrzeby wszystkie zera.
3. **Niekompatybilne interfejsy** - z czysto programistycznego punktu widzenia, często biblioteki do obliczania fingerprintów są przestarzałe, rozwijane przez lata, mają liczne niekompatybilne interfejsy i dług techniczny. Często nie da się ich prosto (lub wręcz w ogóle) użyć w połączeniu z językiem Python i biblioteką scikit-learn, czyli standardowymi narzędziami w uczeniu maszynowym. Jest to szczególnie problematyczne dla chemików czy biologów, którzy typowo nie mają zaawansowanej wiedzy programistycznej.
4. **Uboga dokumentacja** - istniejące biblioteki nie posiadają czytelnej i uporządkowanej dokumentacji, pozwalającej ekspertom na znalezienie potrzebnych im narzędzi i funkcjonalności. Co gorsza, istniejąca dokumentacja często jest przestarzała i błędna, wręcz utrudniając użycie.
5. **Niska jakość wdrożeniowa** - istniejące oprogramowanie często nie realizuje dobrych praktyk programistycznych, jak np. procesy CI/CD, narzędzia do analizy kodu, czy skany bezpieczeństwa.

scikit-fingerprints

Rezultat naszego projektu - biblioteka w języku Python do efektywnego obliczania fingerprintów molekularnych. Rozwiązuje ona problemy istniejącego oprogramowania, i stanowi doskonałe narzędzie do badań i wdrożeń z zakresu chemoinformatyki. Główne cechy:

1. **Równoległość** - fingerprinty są obliczane równolegle, z licznymi optymalizacjami (np. serializacji danych, tworzenia procesów, minibatchingu).
2. **Kompatybilność ze scikit-learn** - jak wskazuje nazwa, w pełni integrujemy się z biblioteką scikit-learn oraz jej API, co zapewnia łatwość użycia i integrację z całym ekosystemem ML.
3. **Duża liczba algorytmów** - zaimplementowaliśmy ponad 30 najpopularniejszych fingerprintów, więcej niż jakiekolwiek inne tego typu narzędzie.
4. **Bogata dokumentacja** - wszystkie fingerprinty mają dokładną dokumentację, z opisami działania, poszczególnych parametrów, oraz bibliografią.
5. **Wysoka jakość kodu** - wykorzystujemy liczne narzędzia do zapewnienia wysokiej jakości kodu, jak kontrola wersji (Git, GitHub), liczne analizy jakości kodu przed każdym commitem (np. black, flake8, pyupgrade), czy narzędzia do analizy bezpieczeństwa i skanowania zależności (np. bandit, safety). Oprogramowanie jest wdrażane z pomocą zautomatyzowanego procesu CI/CD, wykorzystującego GitHub Actions, zawierającego też ponad 200 testów jednostkowych i integracyjnych.
6. **Dodatkowe funkcjonalności** - dodatkowe liczne algorytmy i funkcjonalności ułatwiające badania naukowe dotyczące uczenia na molekułach. Są to np. ładowanie popularnych zbiorów danych, metryki jakości dla problemów wielozadaniowej klasyfikacji, obliczanie konformacji 3D molekuł, czy zoptymalizowany tuning hiperparametrów dla fingerprintów.

Repozytorium GitHub

https://github.com/scikit-fingerprints/scikit-fingerprints

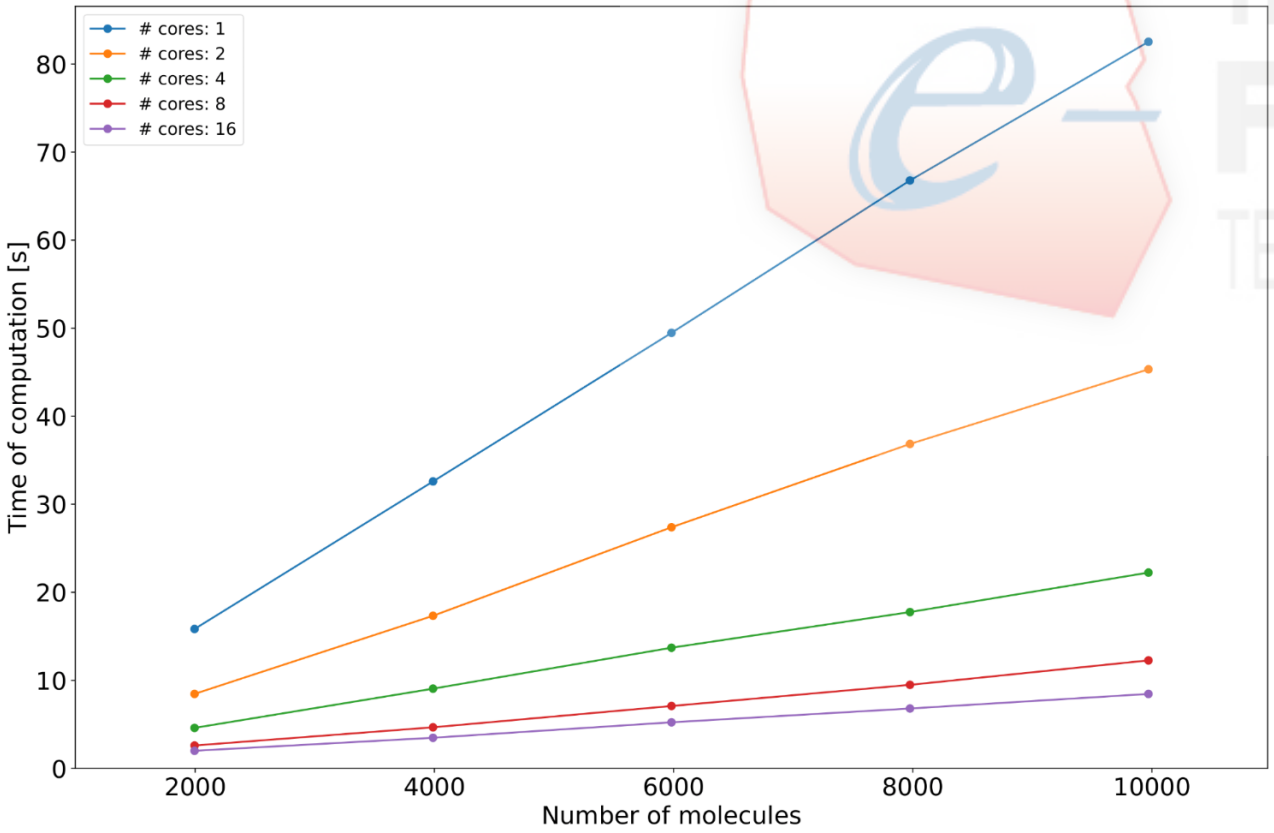
Efektywna współbieżność

Przy użyciu biblioteki joblib, zaimplementowaliśmy **obliczanie współbieżne fingerprintów**. Ze względu na naturę problemu, można tu uzyskać praktycznie liniowe przyspieszenie (*speedup*), a rozwiązanie praktycznie doskonale skaluje się do większych zbiorów danych.

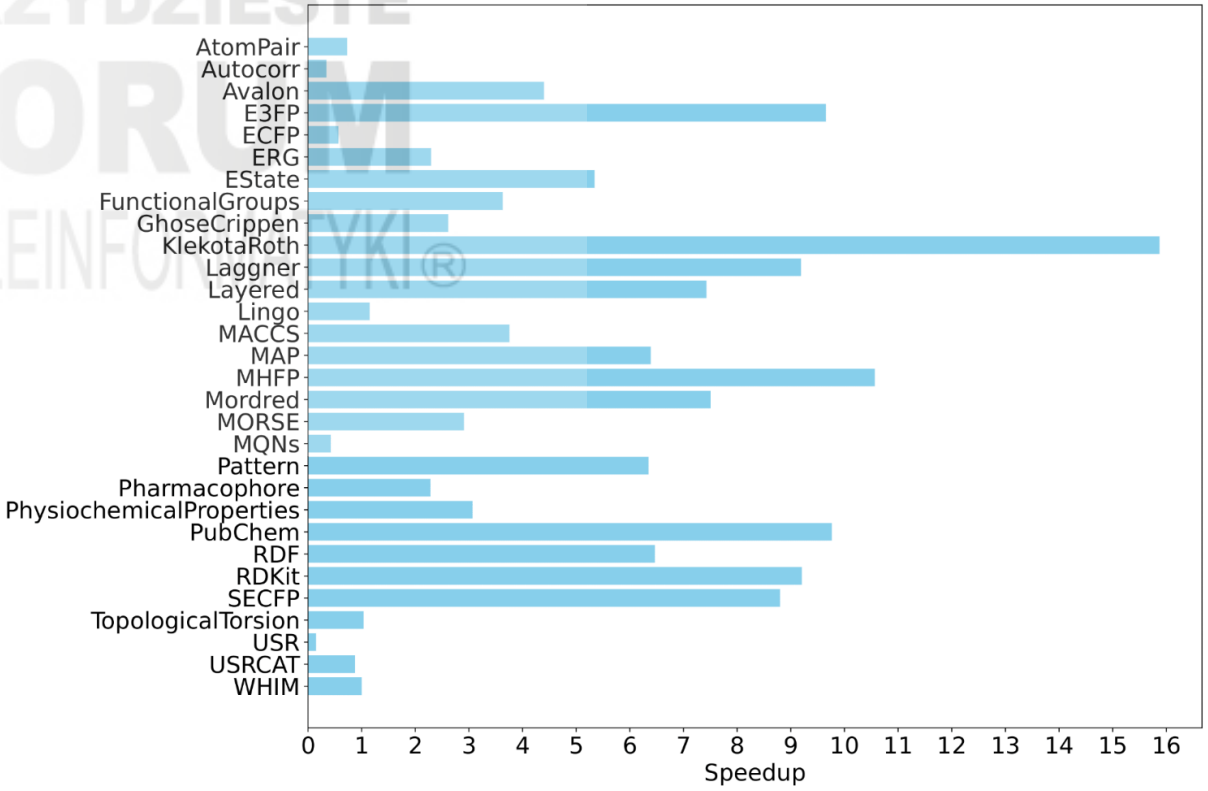
Wspieramy także obliczenia rozproszone z pomocą biblioteki Dask, a obliczenia można w prosty sposób uruchomić na superkomputerach i klastrach HPC. Pozwala to na łatwe obliczenia dużej skali na masywnych zbiorach molekuł, np. w zastosowaniach *high throughput screening*.

Wykonane eksperymenty wskazują jednoznacznie, że fingerprinty znacznie zyskują na obliczeniach współbieżnych:

Czas obliczeń dla wielu rdzeni - fingerprint PubChem



Przyspieszenie (speedup) fingerprintów dla 16-rdzeniowego CPU



Zysk pamięciowy

scikit-fingerprints implementuje **pełne wsparcie macierzy rzadkich**. Pozwala to na redukcję ilości danych przesyłanych między procesami oraz znacznie zmniejsza wynikowe macierze cech. Pozwala to przetwarzać większe zbiory danych oraz redukować koszty, dzięki możliwości wykorzystania mniejszych maszyn.

Przeprowadziliśmy eksperyment dla zbioru PBCA z benchmarku MoleculeNet. Zbiór ten składa się z blisko 440 tysięcy molekuł. Obliczyliśmy wszystkie fingerprinty z naszej biblioteki, a dla podsumowania wyników wybraliśmy 5 tych, dla których uzyskano największą redukcję zużycia pamięci przy zastosowaniu macierzy rzadkich.

Wyniki zaprezentowano poniżej. Redukcje zużycia pamięci są bardzo znaczące, a przy tym należy zaznaczyć, że jest to pojedyncza instancja takiej macierzy. W przypadku równoległej optymalizacji hiperparametrów kopia takiej macierzy musi być przechowywana w pamięci przez każdy proces, więc np. kilkanaście razy. Nie ma to jednocześnie żadnego kosztu czasowego, a jest czasami jest nawet nieco szybsze, bo macierze rzadkie przekazują bardzo mało danych między procesami.

Fingerprint name	Dense array size (MB)	Sparse array size (MB)	Memory savings
Klekota-Roth	2029	23	88.2x
FCFP	855	15	57x
Physiochemical Properties	855	17	50.3x
ECFP	855	19	45x
Topological Torsion	855	19	45x

Przewidywanie własności molekuł

W ramach kolejnych eksperymentów zastosowaliśmy scikit-fingerprints do **tuningu hiperparametrów** w przewidywaniu własności molekularnych. Optymalizacja hiperparametrów jest standardową techniką uczenia maszynowego dla uzyskania lepszych wyników. Co zadziwiające, w literaturze prawie nigdy nie jest ona stosowania w odniesieniu do samych fingerprintów. Mają one jednak liczne parametry, np. długość, wariant binarny lub zliczający poszczególne cechy, czy maksymalne długości rozważanych ścieżek. Prawdopodobnie obecne biblioteki przez niewygodne użycie oraz brak obliczeń równoległych praktycznie uniemożliwiają wykorzystanie takich technik przy rozsądnym nakładzie pracy.

Stworzyliśmy pipeline do klasyfikacji składający się z fingerprintu oraz klasyfikatora Random Forest. Optymalizowaliśmy tylko hiperparametry fingerprintów, aby ewentualny zysk pochodził tylko z tego źródła. Porównaliśmy następnie wyniki bez tuningu oraz z tuningiem hiperparametrów, na 3 zbiorach z benchmarku MoleculeNet: BACE, BBBP oraz HIV. Metryką było Area Under Receiver Operating Characteristic curve (AUROC). Poniżej prezentujemy wyniki dla 5 fingerprintów, które uzyskały najwyższy średni zysk.

Uzyskaliśmy znaczącą poprawę wyników, a przy tym nie wymagało to szczególnie długiego czasu przy wykorzystaniu laptopa o przeciętnej mocy obliczeniowej. Zyski rzędu 5.8% AUROC (BBBP, RDKit fingerprint) są bardzo duże w kontekście praktycznego przewidywania zdolności molekularnych. Ogólnie uzyskane wyniki po tuningu, pomimo prostoty, są bardzo zbliżone do state-of-the-art (SOTA) z literatury.

Fingerprint	Dataset AUROC and tuning gain			Average tuning AUROC gain
	BACE	BBBP	HIV	
GhoseCrippen	84.0 (+2.9)	73.3 (+4.9)	76.0 (+4.3)	+4.0
RDKit	83.0 (+1.2)	73.0 (+5.8)	76.7 (+0.6)	+2.5
Laggner	80.1 (+3.1)	73.8 (+0.7)	76.1 (+1.0)	+1.6
Avalon	83.8 (+2.3)	71.3 (+0.6)	78.0 (+1.7)	+1.5
EState	82.3 (+1.7)	71.7 (+1.0)	76.4 (+0.0)	+0.9

Wpływ na działalność akademicką

Badania i artykuły naukowe

scikit-fingerprints jest używane w licznych badaniach naukowych, oraz zostało użyte w licznych artykułach naukowych oraz wystąpieniach konferencyjnych, np.:

- J. Adamczyk, W. Czech "VMolecular Topological Profile (MOLTOP) - Simple and Strong Baseline for Molecular Graph Classification" European Conference on Artificial Intelligence (ECAI) 2024 (zaakceptowane do publikacji)
- J. Adamczyk, P. Ludynia "Scikit-fingerprints: easy and efficient computation of molecular fingerprints in Python" ArXiv preprint
- J. Adamczyk, J. Poziemski, P. Siedlecki "ApisTox: a new benchmark dataset for the classification of small molecules toxicity on honey bees" (klasyfikacja danych z tego zbioru w ramach kolejnej publikacji)

Prace magisterskie i doktorskie

Biblioteka jest aktywnie używana w ramach prac magisterskich i badań w ramach doktoratów na Wydziale Informatyki AGH, dotyczących przewidywania własności molekuł:

- J. Adamczyk "Supervised learning on graphs"
- P. Ludynia "Benchmarking molecular fingerprints performance and tunability in molecular property prediction"
- M. Stefanik "Molecular similarity graph approach to molecular property prediction"
- M. Praski "Biological and chemical application of graph machine learning"

Projekty

W ramach angażowania studentów w działalność naukową, prowadzone są projekty związane z chemoinformatyką i zastosowaniami uczenia maszynowego na fingerprintach molekularnych.

We współpracy z Kołem Naukowym BIT powstaje projekt implementacji splitów molekularnych, służących do uczciwej ewaluacji algorytmów. W ramach przedmiotów Działalność Naukowa oraz Uczenie Maszynowe prowadzone są projekty dotyczące rozwoju biblioteki o dodatkowe funkcjonalności oraz badanie ich wpływu na jakość klasyfikacji molekuł.