1. Implementacja filtrów molekularnych (1-2 osoby)

Filtry molekularne (ang. *molecular filters*) służą do wstępnego odfiltrowania molekuł uważanych za mało obiecujące. Mogą być zbyt duże, zbyt elastyczne (więc potencjalnie niestabilne), słabo rozpuszczalne etc. Stworzono w związku z tym wiele zestawów warunków, które powinny spełniać molekuły będące potencjalnymi lekami, pestycydami, czy innymi użytecznymi substancjami.

Do tej pory takie podejście było stosowane głównie do metod nienadzorowanych, np. do wstępnego filtrowania ogromnych baz molekuł w pierwotnej fazie wyszukiwania, tzw. *virtual screening*. Z perspektywy uczenia maszynowego takie filtrowanie można by potraktować jako preprocessing danych treningowych, uznając molekuły niespełniające warunków za outliery, błędy, punkty niskiej jakości etc. Takiego filtrowania można by dokonać przed lub po podziale na zbiór treningowy i walidacyjny.

Filtry są z założenia bardzo proste, bo mają być wysoce interpretowalne. Projektuje się je statystyką (oraz czasem prostym ML) na ogromnych bazach molekuł, żeby odwzorować ogólne wzorce. Ich implementacja to zwykle kilka zasad if/else dla prostych cech molekuł.

Przydatne tutoriale o filtrowaniu molekuł:

- https://projects.volkamerlab.org/teachopencadd/talktorials/T002_compound_adme.html
- https://projects.volkamerlab.org/teachopencadd/talktorials/T003_compound_unwanted_substructures.html
- https://github.com/sriniker/TDT-tutorial-2014/blob/master/TDT_challenge_tutorial.html

Elementy projektu:

- zaimplementować brakujące filtry molekularne w bibliotece scikit-fingerprints, za pomocą biblioteki RDKit, lista jest w tym issue
- przetestować, czy filtrowanie (samych danych treningowych przy holdout, lub train+valid z użyciem walidacji skrośnej) poprawia wyniki klasyfikacji, np. na benchmarkach MoleculeNet (z pomocą OGB) lub TDC
- przykład klasyfikacji
- można też wypróbować, czy luzowanie filtrów (dopuszczanie 1 violation) lub łączenie filtrów poprawia wyniki

Oczekiwane wyniki:

- zaimplementowane filtry molekularne
- wyniki klasyfikacji na benchmarkach
- stwierdzenie, czy filtrowanie poprawia wyniki

UWAGA: Max. 1 zespół projektowy

2. Implementacja deskryptorów molekularnych (2-3 osoby)

Istnieją algorytmy wektoryzujące od razu całe molekuły, czyli fingerprinty molekularne (ang. *molecular fingerprints*). Zamiast tego można dokonać ręcznej inżynierii cech, czyli deskryptorów molekularnych. Są to pojedyncze wartości (np. Zagreb index), lub zestawy wartości (np. Chi indices), które opisują własności fizykochemiczne molekuły. Chemicy używają ich od dziesięcioleci, w połączeniu ze statystyką (typowo zwykłą regresją liniową) do analizowania molekuł.

Mało kto używa ich obecnie do ML na molekułach, chociaż nie ma szczególnie ku temu przesłanek. Mają często mocne i dobrze udowodnione korelacje z wieloma własnościami fizykochemicznymi molekuł. Ponadto można przyjąć podejście "na masę", czyli obliczamy dużo deskryptorów, a potem robimy agresywną selekcję cech. Jest to możliwe, bo są one bardzo proste i szybkie w obliczaniu.

Można by też zweryfikować, czy istnieje zestaw uniwersalnie przydatnych deskryptorów, dla wielu zadań. Kiedy mamy bardzo dużo zbiorów danych, i deskryptory dla nich wszystkich, to można zrobić średni feature importance dla wszystkich zbiorów danych (uśredniając wagę cechy po wszystkich zbiorach). Tworzyłoby to de facto nowy fingerprint molekularny.

Bardzo dobrze opisane zestawy różnych deskryptorów molekularnych ma biblioteka ChemoPy. Pliki "Descriptor list.pdf" i "manual.pdf" zawierają pogrupowane deskryptory, z opisami i wzorami.

Elementy projektu:

- zaimplementować wybrane zestawy deskryptorów molekularnych w bibliotece scikit-fingerprints, za pomocą biblioteki RDKit
- zasadniczo trzeba po prostu włączyć zestawy deskryptorów z ChemoPy
- trzeba zwrócić uwagę na efektywną implementację, np. często wiele deskryptorów używa najkrótszych ścieżek, więc można tego reużyć dla poszczególnych cech z wektora
- przetestować jakość poszczególnych deskryptorów, np. na benchmarkach MoleculeNet (z pomocą OGB) lub TDC
- przykład klasyfikacji
- sprawdzić, czy da się znaleźć średnio dobry zestaw deskryptorów, wykorzystując średni wynik walidacyjny na wielu zbiorach

Oczekiwane wyniki:

- zaimplementowane deskryptory molekularne
- wyniki klasyfikacji na benchmarkach
- stwierdzenie, czy da się stworzyć "średnio dobry" zestaw deskryptorów molekularnych

3. Frequent subgraph mining w klasyfikacji molekuł (1-2 osoby)

Frequent pattern mining, typowo używany w analizie i eksploracji danych, można też zastosować do grafów, i nazywamy go wtedy frequent subgraph mining (FSM). W szczególności można wyróżnić wariant discriminative FSM, w którym staramy się wydobyć takie fragmenty, które są częste w klasie pozytywnej, a rzadkie w negatywnej.

Robi się to w szczególności dla grafów molekularnych, bo dla nich wiadomo, że częste subgrafy są ważne (np. grupy funkcyjne). Nikt raczej nie próbował jeszcze używać podejścia FSM do klasyfikacji grafów. Jest to de facto stworzenie data-driven zestawu cech, dla konkretnego zbioru danych, który powinien dobrze rozróżniać klasy. Jest to z jednej strony bardzo interpretowalne, a z drugiej bardziej elastyczne niż podobnie działające tzw. *substructure-based molecular fingerprints* (np. MACCS, Klekota-Roth, PubChem).

Doskonałym i popularnym narzędziem tutaj jest MoSS (Molecular SubStructure miner). Wspiera on wszystkie kluczowe parametry FSM, w tym discriminative FSM. Wadą jest natomiast, że to typowy program w Javie, mający plikowe I/O. Aby użyć go w Pythonie, trzeba go uruchomić z poziomu CLI, oraz odpowiednio procesować pliki.

Elementy projektu:

- napisać klasę ekstrahującą cechy z pomocą MoSS, zgodną z interfejsem scikit-learn (z odpowiednimi hiperparametrami)
- przetestować jakość takiego podejścia, np. na benchmarkach MoleculeNet (z pomocą OGB) lub TDC
- przykład klasyfikacji
- wypróbować bez tuningu hiperparametrów, tune'ując poszczególne hiperparametry z osobna, oraz wszystkie razem

Oczekiwane wyniki:

- zaimplementowany fingerprint molekularny z pomocą MoSS
- wyniki klasyfikacji na benchmarkach
- stwierdzenie, czy takie podejście daje dobre wyniki

4. Klasyfikacja peptydów za pomocą metod molekularnych (2-3 osoby)

Peptydy to małe białka, pełniące kluczowe funkcje w żywych organizmach. Co więcej, jak każde białka pełnią często wiele funkcji naraz, które wpływają na ich bardziej wysokopoziomowe cechy. Ze względu na relatywnie mały rozmiar, można je przetwarzać rozsądnym kosztem obliczeniowym, który jest często problemem dla większych białek.

Do tej pory nie próbowano raczej na większą skalę traktować peptydów po prostu jako grafów molekularnych. Zamiast tego używa się dedykowanych algorytmów, które typowo operują na sekwencji aminokwasów. Pytanie, czy użycie fingerprintów molekularnych (ang. *molecular fingerprints*) do wektoryzacji molekuł, które operują wprost na grafach molekuł, da dobre wyniki. Jest to bardzo niskopoziomowa i szczegółowa reprezentacja.

Zbiorów z klasyfikacji (i regresji) peptydów jest dużo, np.:

- 2 zbiory z Long Range Graph Benchmark
- 3 zbiory HemoPI
- liczne zbiory peptydów antybakteryjnych (AMPs)
- zbiór pestycydowych AMPs (paper)
- cholesterol-lowering peptides (dane stąd)
- bioactive peptides

Elementy projektu:

- zebranie zbiorów do klasyfikacji peptydów
- przetestowanie fingerprintów molekularnych z biblioteki scikit-fingeprints do klasyfikacji peptydów
- porównanie np. z ProtBERT albo deskryptorami z PyBioMed

Oczekiwane wyniki:

- benchmark fingerprintów do klasyfikacji peptydów
- porównanie z ProtBERT i deskryptorami białkowymi

UWAGA: Max. 1 zespół projektowy

5. Benchmark kerneli grafowych w klasyfikacji molekuł (2-4 osób)

Metody kernelowe jak np. kernel SVM, kernel PCA, kernel ridge regression i inne działają dla dowolnych obiektów. Tak naprawdę nie przyjmują one na wejściu wierszy macierzy cech kształtu N*D, tylko macierz podobieństw punktów kształtu N*N.

Jak mamy taką macierz, kernel SVM po prostu zadziała, i szczególnym przypadkiem obiektów tutaj są grafy. Definiuje się dla nich kernele grafowe. Operują one na cechach topologicznych grafów, starając się zmierzyć ich podobieństwo strukturalne. Takie podejście jest często bardzo elastyczne, uwzględniając bogatą strukturę podgrafów, i dobrze działa dla małych zbiorów danych. Kernele grafowe, w zależności od typu, mogą ponadto uwzględniać dyskretne node labels (np. typ atomu) i edge labels (np. rodzaj wiązania), jak i ciągłe node attributes (np. ładunek atomu) i edge attributes (np. długość wiązania).

W ostatnich latach ze względu na popularność grafowych sieci neuronowych (GNNs), kernele grafowe bardzo straciły na popularności. Wyniki ze starych prac wskazują jednak, że dalej mają bardzo mocny performance, a brak podawania ich wyników to zasadniczy błąd. Solidny benchmark do klasyfikacji molekuł prawdopodobnie pokaże, że ich wyniki potrafią być blisko state-of-the-art dla małych zbiorów danych, które są w chemoinformatyce bardzo częste.

Ważne jest tutaj, że obliczanie niektórych kerneli jest dość czasochłonne. Samą macierz kerneli można obliczyć raz dla danego zbioru treningowego, a potem reużywać w walidacji skrośnej - wystarczy wycinać odpowiednie wiersze i kolumny. Hiperparametry ma i sam graph kernel, i SVM (siła regularyzacji), i warto to przy walidacji skrośnej optymalizować w dwóch krokach: liczymy kernel dla danych hiperparametrów, a potem tune'ujemy C. Scikit-learn niestety tego nie implementuje, więc trzeba by to napisać samemu.

Elementy projektu:

- zbenchmarkować kernele grafowe do klasyfikacji grafów, np. na benchmarkach MoleculeNet (z pomocą OGB) lub TDC
- kernele grafowe implementuje biblioteka GraKeL, inne ciekawe to np. P-WL, WWL, FGW, graphit-learn, karateclub (np. GL2Vec, Graph2Vec)
- dokonać tuningu hiperparametrów dla kerneli, z zakresami wziętymi z paperów
- można też rozważyć przybliżanie kerneli z pomocą Nyström approximation (tutorial, dyskusja)
- porównać wyniki z leaderboardami

Oczekiwane wyniki:

- benchmark kerneli grafowych dla klasyfikacji molekuł

6. Klasyfikacja szeregów czasowych dla małych danych (2-3 osoby)

Problem klasyfikacji szeregu czasowego jest typowy dla danych będących sygnałami, np. z urządzeń medycznych, akcelerometrów, smart watchy. W szczególności można ich użyć do diagnozy chorób objawiających się zaburzeniami ruchu, głównie neurologicznych oraz psychicznych.

Są tutaj na przykład ciekawe zbiory danych jak Depresjon, Psykose czy HYPERAKTIV. Zawierają one jednowymiarowy szereg czasowy z aktywnością dobową pacjenta, mierzoną z dość dużą częstotliwością przez kilka-kilkanaście dni. Problemem jest jednak mała liczba pacjentów - zaledwie ok. 50 (lub 100 dla ostatniego zbioru o ADHD), a to właśnie ich klasyfikujemy. Jest to problem i na etapie uczenia (jak nie przeuczyć), jak i testowania (jak ewaluować zdolność generalizacji).

Niektóre papery dla takiego problemu uczenia na małych danych wykorzystują tzw. multiple instance learning (MIL). W takim przypadku wycinamy poszczególne dni, dostają one taką klasę jak pacjent, i uczymy na takich kawałkach. Klasyfikacja per pacjent wymaga agregacji predykcji dla dni, np. "jeżeli większość dni to depresja, to diagnozuj depresję". Jak robić taką agregację, to ważny hiperparametr. Trzeba tutaj jednak uważać z ewaluacją - ona dalej jest na całych pacjentach (to wiele paperów robi źle).

Bazą projektu jest artykuł "Comparison of Manual and Automated Feature Engineering for Daily Activity Classification in Mental Disorder Diagnosis" J. Adamczyk, F. Malawski, oraz artykuły poszczególnych zbiorów danych.

Elementy projektu:

- dokonać klasyfikacji na 3 zbiorach danych
- można wykorzystać procedurę ewaluacji nested CV z powyższego artykułu
- zweryfikować, czy podejście MIL przy poprawnej ewaluacji coś daje w porównaniu do prostej klasyfikacji całych danych
- sprawdzenie, czy np. same dni/noce działają lepiej od całych danych (zastosowane w moim paperze powyżej
- podstawową biblioteką tutaj jest sktime, oraz metody np. Dynamic Time Warping (DTW), Time Series Forest, ROCKET, shapelet transform, TSFRESH

Oczekiwane wyniki:

- klasyfikacja poszczególnych zbiorów danych
- porównanie klasyfikacji całości szeregu oraz poszczególnych dni (MIL)
- porównanie użycia całych danych, samych dni oraz samych nocy
- benchmark poszczególnych algorytmów

7. Implementacja metryki odległości dla mieszanych typów cech (2-3 osoby)

Praktycznie wszystkie metryki odległości działają tylko dla zmiennych numerycznych, lub tylko binarnych, tylko numerycznych etc. Większość realnych danych ma zmienne o typach mieszanych, i tradycyjnie przekształca się wszystko na zmienne numeryczne. Minusem jest, że techniki takie jak one-hot encoding bardzo zwiększają wymiarowość i zmniejszają interpretowalność.

Istnieją metryki dedykowane stricte danym o typach mieszanych, z których najbardziej znana jest metryka Gowera (Gower's metric). Polega na połączeniu metryk dla różnych typów danych, i ma wszystkie pożądane matematyczne własności. Wstępna wersja jest gotowa - https://github.com/Arch4ngel21/GowerMetric, a teraz trzeba to rozbudować do pełnej Pythonowej biblioteki.

W szczególności warto dodać też wsparcie dla zmiennych kategorycznych uporządkowanych (categorical ordinal), tzw. Podani's modification. Co ciekawe, metryka Gowera może też tworzyć unikatowy kernel dla SVMów (link poniżej). Istnieją też inne metryki dla danych mieszanych, z którymi można się porównać, dobry przegląd ma "A survey of distance measures for mixed variables" S. Bishnoi, B. Hooda (link).

Są główne 3 benchmarki do danych tabelarycznych, podlinkowane i dobrze opisane tutaj: https://youngandbin.medium.com/tabular-benchmark-dataset-aea2c09a8fef

Przydatne źródła:

- "Distances with Mixed-Type Variables, some Modified Gower's Coefficients" M. D'Orazio - link
- prezentacja z konferencji
- https://stats.stackexchange.com/questions/15287/hierarchical-clustering-with-mixed-type-data-what-distance-similarity-to-use
- metryka Gowera jako kernel
- "A General Coefficient of Similarity and Some of Its Properties" J. Gower
- "Extending Gower's general coefficient of similarity to ordinal characters" J. Podani

Elementy projektu:

- dokończyć implementację metryki Gowera, wliczając modyfikację Podaniego
- wdrożyć prosty proces CI/CD z pomocą GitHub Actions i Poetry, można wzorować się na scikit-fingerprints
- porównanie różnych wariantów metryki Gowera z klasycznymi metrykami dla klasyfikacji kNN i klasteryzacji
- implementacja i porównanie z innymi metrykami dla danych mieszanych

Oczekiwane wyniki:

- pełnoprawna biblioteka do obliczania metryki Gowera
- benchmark i porównanie metryki Gowera z klasycznymi miarami odległości
- implementacja innych metryk odległości dla typów mieszanych i porównanie

UWAGA: Max. 1 zespół projektowy