Probabilistyczne Modele Grafowe

Końcowe zadanie projektowe

1 Przed przystąpieniem do realizacji zadania

a. ustalić skład grup
   - grupy 2-3–osobowe

b. wybrać zbiór danych, który będzie służył do analiz w grupie
   - przykładowo z: `https://archive.ics.uci.edu/ml/datasets.php`
   - kryteria wyboru zbioru danych:
     - minimalna liczba cech: 8
     - minimalna liczba instancji: 300

c. grupy ustalają, uzupełniają w formularzu i wysyłają do akceptacji (temat musi być zaakceptowany przed 31.05.2023 r.):
   - nazwę wybranego zbioru danych, na którym będą pracowali (oraz link do tego zbioru),
   - krótki opis zadania/problemu, który będzie rozważany w projekcie – można sugerować się "Default Task" z opisu zbioru danych, ale wskazana jest kreatywność,
   - wybrany sposób ewaluacji modeli i porównania ich wyników,

2 Realizacja zadania

W ramach zadania do zrealizowania jest część programistyczno-analityczna oraz dokumentacyjna. Wszystkie wykonane zadania programistyczno-analityczne muszą być opisane w jednoznaczny sposób w części dokumentacyjnej.

2.1 Przeprowadzenie eksploracyjnej analizy danych

Zastosować wybrane z poniższych propozycji, który jest adekwatny do zbioru i problemu, uzasadnić:

- opis zmiennych (typ, zakres wartości/rozkład, opis słowny),
- zależności zmiennych parami (correlation, scatter plots, box plots, violin plots, mosaic plots, heat maps),
- obserwacje odstające,

- preprocessing (normalizacja, selekcja cech, ekstrakcja cech, zmiana wymiarowości),

2.2 Implementacja modeli

Należy zaimplementować i zbadać 2 lub 3 modele (w zależności od liczności grupy) spośród tych poznanych na zajęciach (wykład + laboratorium). Uzasadnić wybór modeli do danego problemu.

2.3 Przeprowadzenie badania zaimplementowanych modeli na wybranym zbiorze danych

a. ustalenie scenariusza eksperymentalnego (implementacja i opis procedur uczenia i przetwarzania danych, np. 15% danych zachowane do testowania modeli bądź cross-validation itd.)

b. wybór metryk do ewaluacji modeli

c. przegląd hiperparametrów modeli (rozsądny)

d. wizualizacja wyników

e. wnioski

2.4 Raport z realizacji zadania

a. raport w postaci PDF'a, preferowane narzędzie: LaTeX, język: polski

b. raport wraz z całościowy kodem musi być umieszczony na grupowym repozytorium zadania do godziny 22:00 dnia poprzedzającego termin oddania zadania

c. rozdziały raportu (wraz z punktacją do oceny):

   i. **Eksploracyjna analiza danych** (3 pkt.)

   ii. **Modele** (4 pkt.) – wyjaśnienie doboru modeli

   iii. **Eksperymenty**

   - zastosowanie modeli probabilistycznych (8 pkt)
   - porównanie wyników wybranych modeli (3 pkt) - Przeprowadzić tam gdzie jest to możliwe analizę **credible intervals** na poziomie ufności 0.9.

     Dodatkowo:

     – dla klasyfikacji - miary typu f1 score, auc, precision oraz recall (przy stosowaniu istniejących implementacji wziąć pod uwagę implementację typu "micro") z wartości oczekiwanej wielu realizacji modelu

     – dla regresji - RMSE, SSE, $R\hat{2}$, z wartości oczekiwanej wielu realizacji modelu

- dla klasteryzacji - silhouette coefficient, rand index, jaccard index, f-measure, z wartości oczekiwanej wielu realizacji modelu
- dla jakości modeli czysto generatywnych - negative log likelihood

2.5 Kod i reprodukowalność eksperymentów (2 pkt.)

Efektem tego zadania powinien być również kod źródłowy zaimplementowanych modeli probabilistycznych oraz przeprowadzonych eksperymentów. Należy zadbać o jakość i czytelność tego kodu.

W ramach dobrych praktyk procesu data science, cały scenariusz eksperymentalny powinien być w pełni odtwarzalny (reprodukowalność eksperymentów), tzn. powinno być możliwe uruchomienie eksperymentów przez prowadzących i otrzymanie takich samych wyników jak w złożonym raporcie. W tym celu można wykorzystać różne podejścia, m.in. **skrypty powłoki Bash** ("skrypty shellowe"), które będą uruchamiać skrypty Pythonowe w odpowiedniej kolejności z właściwymi argumentami. Podobne rozwiązanie można uzyskać wykorzystując tzw. **Makefile**.

Narzędziem o którym warto wspomnieć jest również **DVC (Data Version Control)**, które służy właśnie do wersjonowania kodu wraz z danymi (wykorzystując system kontroli wersji git oraz zewnętrzne repozytoria danych - przy czym te drugie są nieobowiązkowe). Definiowane są tutaj DVC stages, które odpowiadają kolejnym etapom w przetwarzaniu danych (czyszczenie danych, wizualizacja, ekstrakcja cech, uczenie modeli, ewaluacja modeli, agregacja wyników, wizualizacja wyników itp.)