Algorytmiczna Analiza Danych 2024/2025
Lista zadań na laboratoria

1 Python + Jupyter Notebook (deadline: 2gie laboratoria)

Zakładam, że znasz podstawy Pythona. Zadania w tej sekcji przeprowadzą Cię przez proces tworzenia wygodnego środowiska programistycznego. Na końcu tej sekcji powinieneś być w stanie pisać i uruchamiać kod Python w Jupyter Notebook oraz znać podstawowe polecenia Jupyter Notebook. Powinieneś również wiedzieć, jak instalować nowe pakiety i przełączać środowiska.

Zadanie 1 — (0p) Pobierz i zainstaluj Anacondę. Ewentualnie, jeśli masz już zainstalowanego Pyhtona i kluczowe w analizie danych pakiety, zainstalowanie Jupyter Notebook jest wystarczające. Przejrzyj przewodnik użytkownika oraz ściągawkę conda, aby nauczyć się, jak tworzyć i przełączać środowiska oraz instalować nowe pakiety.

Zadanie 2 — (0p) Uruchom Jupyter Notebook i zobacz *Help → User Interface Tour, Keyboard Shortcuts*. Naucz się tworzyć, przesuwać i uruchamiać komórki. Naucz się uzyskiwać informacje o obiektach i metodach oraz jak wyświetlać ich kod źródłowy (np. możesz użyć *tab, shift+tab, shift+tab+tab lub wpisać '?' i '??' przed nazwą metody*).

Zadanie 3 — (0p) Przypomnij sobie, jakie struktury danych są dostępne w Pythonie. Zwróć uwagę na list comprehension, często pomagają pisać czytelny i wydajny kod.

Zadanie 4 — (0p) Zainstaluj i zapoznaj się z pakietami `numpy`, `pandas`, `matplotlib`.

a) Przeczytaj oficjalny przewodnik użytkownika NumPy.

b) Przeczytaj oficjalny przewodnik użytkownika pandas.

c) Przeczytaj oficjalny przewodnik użytkownika matplotlib.

Zadanie 5 — (1p) Znajdź źródło swojej ulubionej książki i zapisz je w formacie UTF-8. Wczytaj książkę i podziel ją na pojedyncze słowa. Zamień wszystkie litery na małe litery, usuń znaki interpunkcyjne i usuń słowa nieistotne (ang. stop words). Listę słowa nieistotnych możesz znaleźć w Internecie (np. tutaj). Możesz również spróbować użyć stemmingu lub lematyzacji, aby zredukować różne formy danego słowa do wspólnej formy:

```
from stemming.porter2 import stem
filtered_words = [stem(word) for word in filtered_words]
```

Następnie przekształć uzyskaną listę słów w listę par `(word,1)` typu `(String,Int)`:

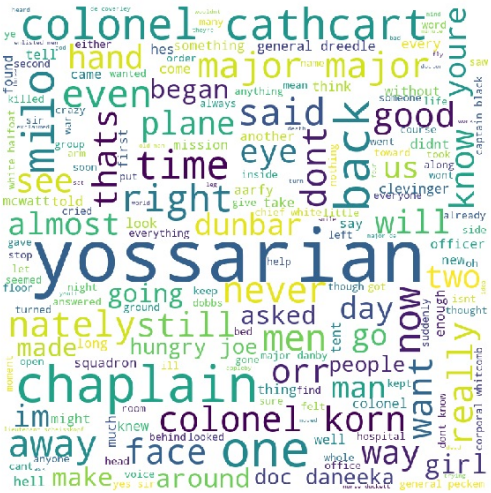
```
pairs = [(w,1) for w in filtered_words]
```

Pogrupuj listę par według różnych słów i policz łączną liczbę wystąpień każdego słowa, aby uzyskać pary `(word, occurrences)`. Na przykład możesz użyć metody `groupby`. Jednak

zauważ, że metoda `groupby` wymaga, aby lista wejściowa była posortowana według kluczy, a takie sortowanie może być kosztowne obliczeniowo dla dużych list. Spróbuj wymyślić i zaimplementować bardziej efektywny sposób agregacji wystąpień, który nie wymaga sortowania.

```
from itertools import groupby
pairs.sort()
word = lambda pair : pair[0]
grouped_pairs = [(w, sum(1 for _ in g)) for w, g in groupby(pairs,
                                                key=word)]
```

Posortuj uzyskaną listę par według drugiego elementu w kolejności malejącej. Usuń kilka początkowych elementów posortowanej list (mało istotne, najczęściej pojawiające się słowa) i zapisz wynik do pliku tekstowego. Zbuduj chmurę słów z uzyskanej listy. Możesz użyć biblioteki Pythona lub serwisu http://www.wordclouds.com/.



Zadanie 6 — (1p) To jest kontynuacja poprzedniego zadania.

1. Podziel swoją książkę na rozdziały. Traktujemy każdy rozdział jako osobny dokument.
2. Podziel każdy dokument na słowa (używając małych liter, lematyzacji itp.).
3. Wyznacz wagi tf-idf dla wszystkich słów we wszystkich dokumentach:

$$tf\text{-}idf(t,d,D) = tf(t,d) \times idf(t,D),$$

gdzie t oznacza termin (słowo), d oznacza dokument, a D oznacza zbiór wszystkich dokumentów. Częstotliwość słowa $tf(t,d)$ to liczba wystąpień słowa t w dokumencie d. Odwrotna częstotliwość dokumentu $idf(t,D)$ jest często definiowana jako

$$idf(t,D) = \log \frac{|D|}{1 + |\{d \in D : t \in d\}|}.$$

Uzasadnij sensowność powyższej formuły. Istnieją pakiety, które ułatwiają obliczanie wag tf-idf, ale spróbuj zaimplementować odpowiednią procedurę samodzielnie.

4. Dla każdego dokumentu osobno zbuduj chmurę słów używając uzyskanych wag tf-idf.

Zadanie 7 — (1p) Napisz funkcję, która jako wejście przyjmuje słowo i używa wag tf-idf do stworzenia listy rozdziałów twojej książki najbardziej pasujących do tego słowa (tj. powinna zwrócić listę rozdziałów posortowanych według odpowiednich wag tf-idf).

Zadanie 8 — (1p) Dla każdego danego słowa w swojej książce utwórz listę pięciu najczęstszych słów, które pojawiają się bezpośrednio po rozważanym słowie (ale pomiń słowa stopu). Użyj tego podsumowania, aby wygenerować losowy akapit, który przypomina akapit z twojej książki.

2 Regresja liniowa (deadline: 3cie laboratoria)

Zadanie 9 — (1p) Rozwiąż zadanie 13 z Sekcji 3.7 w ISL.

Zadanie 10 — (1p) Rozwiąż zadanie 14 z Sekcji 3.7 w ISL.

Zadanie 11 — (5p) Pobierz plik Auto.csv. Wczytaj go jako DataFrame i zmień kolumnę `origin` na typ `category`. Podziel dane na zbiór treningowy i walidacyjny.

a) Użyj biblioteki *statsmodels* do wykonania regresji liniowej z `mpg` jako zmienną objaśnianą, a `horsepower` jako predyktorem (np. metoda OLS). Wyjaśnij informacje zwracane przez metodę `model.summary()`, w szczególności: przedziały ufności, p-values, T-statistic, F-statistic i R^2. Możesz się wzorować na tej analizie w Jupyter notebook.

b) Utwórz macierz wykresów punktowych (ang. scatterplot matrix), dla wszystkich zmiennych w zbiorze danych. Możesz użyć `pandas.plotting.scatter_matrix(...)`. Oblicz macierz korelacji między zmiennymi. Możesz użyć funkcji `corr()` dla DataFrame z biblioteki *pandas*.

c) Przeprowadź regresję liniową z `mpg` jako zmienną objaśnianą i wszystkimi innymi zmiennymi (z wyjątkiem `name`) jako predyktorami. Spróbuj zdefiniować różne modele wykorzystując biblioteką *patsy* i używając symboli `+`, `*`, `:` oraz różnych transformacji zmiennych, takich jak na przykład `I(np.log(X))` lub `I(np.sqrt(X))`. Co to jest błąd generalizacji? Dla którego modelu uzyskujesz najlepszy błąd generalizacji?

d) Spróbuj znaleźć wartości odstające (ang. outliers) i usuń je ze zbioru danych. Możesz wykorzystać np. wykres reszt (ang. residual plot) albo Z-Score. Czym są punkty o wysokiej dźwigni (ang. high leverage points)? Jak możesz je wykryć (np. zobacz tutaj)? Ponownie wytrenuj swoje modele na oczyszczonych danych i porównaj wyniki. Pamiętaj, że usuwając obserwacje ze zbioru danych powinieneś mieć silne przesłanki, że zawierają one nieprawidłowe informacje. Usuwanie obserwacji tylko dlatego, że nie są podobne do innych nie jest dobrym pomysłem.

3 Lista trzecia (deadline: 4te laboratorium)

Zadanie 12 — (1p) Używając danych z pliku Auto.csv z poprzedniej listy, utwórz i porównaj dwa modele regresji liniowej do przewidywania wartości *mpg*. W pierwszym modelu użyj *year* traktowanego jako zmienna ciągła. W drugim modelu użyj *year* traktowanego jako zmienna kategoryczna. Który model jest lepszy? Co jeśli byłoby znacznie więcej niż 13 wartości zmiennej *year*? Możesz zobaczyć przykłady w tym Jupyter notebook.

Zadanie 13 — (10p) Pobierz plik Credit.csv opisany tutaj. Twoim zadaniem jest znalezienie modelu, który będzie miał możliwie dużą skuteczność przewidywania:

a) czy dana osoba ma dochód większy niż 50 (wskazówka: stwórz nową zmienną indykatorową na bazie *Income* i usuń zmienną *Income*),

b) ile kart kredytowych posiada dana osoba.

Rozważ różne modele, np. regresje logistyczną, drzewa decyzyjne, lasy losowe, k-najbliższych sąsiadów, oraz różne parametry modeli (np. maksymalna głębokość drzewa, różne wartości k najbliższych sąsiadów). Możesz wykorzystać implementację tych modeli np. z pakietu scikit-learn

Jaką dokładność (ang. accuracy) ma najlepszy model, który udało Ci się uzyskać w przypadku a) i b)? Aby uzyskać wiarygodną odpowiedź użyj walidacji krzyżowej do przeprowadzenia wielu eksperymentów w oparciu o jeden zbiór danych. Np. użyj metody KFold. Narysuj wykresy typu box-plot, aby wygodnie porównywać modele, uwzględniając nie tylko średnią dokładność ale również wariancję wyników. Możesz zobaczyć przykłady w tym Jupyter notebook.

Zadanie 14 — (5p) Dla problemów a) i b) z Zadania 13 wybierz dwa ciągłe predyktory, które wydają się być istotne, i narysuj granice decyzyjne dla testowanych przez Ciebie modeli. Możesz oprzeć się na tym tutorialu. W punkcie b) mamy do czynienia z więcej niż dwoma klasami, więc do generowania wykresów należy użyć więcej niż dwóch kolorów. Czy jesteś w stanie zaproponować sensowne granice decyzyjne dla problemu z punktu b)?

J.L.