Wstęp do Sztucznej Inteligencji
Laboratorium 3 - Naive Bayes i klasyfikacja tekstów

mgr inż. Andrii Shekhovtsov

26 marca 2025

Zasady oceniania

Program, który powstał w ramach tego zadania, powinien zostać przesłany za pośrednictwem Moodle jako plik tekstowy z kodem w Python w formacie `.py`. W przesyłanym pliku z kodem proszę umieścić na pierwszej linii komentarz ze swoim imieniem, nazwiskiem, numerem albumu oraz numerem grupy. Plik proszę nazwać `wdsi_lab3.py`.

Plik ten należy przesłać za pośrednictwem systemu Moodle w wyznaczonym tam terminie.

W przesyłanym pliku z kodem proszę umieścić na pierwszej linii komentarz ze swoim imieniem, nazwiskiem i numerem albumu.

Przykładowe formatowanie pliku:

```
1 # Jan Kowalski, nr. alb. 12345
2
3 # tutaj umieszczamy cały kod programu...
```

UWAGA: Termin oddania zadania jest ustawiony w systemie Moodle. W przypadku nieoddania zadania w terminie, uzyskana ocena będzie zmniejszana o 0,5 za każdy zaczęty tydzień opóźnienia. Zadania oddawane później niż miesiąc po terminie ustawionym na Moodle mogą zostać niesprawdzone lub ocenione na ocenę niedostateczną.

UWAGA: W przypadku wysłania zadania w formie niezgodnej z opisem w instrukcji prowadzący zastrzega prawo do wystawienia oceny negatywnej za taką pracę. Przykład: wysłanie `.zip` lub `.pdf` tam, gdzie był wymagany plik tekstowy z rozszerzeniem `.py`.

1 Zadania do wykonania

1. Wczytaj zawartość pliku `spam_prepared.csv`[1] w dowolny znany sposób. Plik zawiera dwie kolumny: `is_spam` oraz `text`. Kolumnę `is_spam` należy przerobić tak, by powstał `np.array`, zawierające **liczbowe** wartości $\{0, 1\}$, natomiast kolumnę `text` przerabiamy na `np.array` zawierający napisy.

2. Podziel dane na dane uczące i testowe w taki sposób, że pierwsze 5000 wpisów będą traktowane jako dane uczące, a pozostałe (572) jako dane testowe. W ten sposób powstaną nam 2 zestawy danych: każdy zawiera array z treścia wiadomości oraz array z ich etykietami (0 i 1), jeden ma długość 5000, drugi 572.

   Uwaga: W praktyce do tego najczęściej stosowana jest funkcja `train_test_split` z biblioteki `scikit-learn`, która przeprowadza losowy podział na dane uczące i testowe w zadanej proporcji. Natomiast dane w pliku csv są przygotowane tak że możemy dokonać po prostu odcięcia części danych - proporcje spam/nie spam w danych zostaną zachowane.

3. Przygotuj funkcje do ekstrakcji cech, która przygotuje dwie struktury danych: macierz cech oraz słownik, umożliwiający zamianę tekstu na cechy. Funkcja powinna przyjmować `np.array` (np. *data*) z tekstami wiadomości. Funkcja powinna wykonać następujące kroki:

   (a) Utwórz zbiór zawierający wszystkie słowa we wszystkich dokumentach i posortuj go.

   (b) Utwórz słownik $word\_to\_feature$ który zmapuje każde słowo na indeks tego słowa w posortowanym zbiorze.

[1] Jest to przerobiony dataset z Kaggle (https://www.kaggle.com/datasets/vishakhdapat/sms-spam-detection-dataset)

(c) Utwórz macierz cech o rozmiarze liczba wiadomości ($len(data)$) na liczbę wszystkich słów. Pod indeksem i, j będziemy umieszczać informacje czy j-te słowo występuje w i-tym dokumencie. Zalecany sposób na utworzenie macierzy cech: `np.zeros((len(data), len(word_to_feature)), dtype='int8')`.

(d) Dla każdego tekstu $data[i]$ w zbiorze wykonaj: dla każdego słowa w w tekście $data[i]$ pobierz jego numer cechy $j = word\_to\_feature[w]$ i umieść w macierzy cech 1 pod indeksem $[i, j]$.

(e) Na końcu funkcja powinna zwrócić macierz cech i słownik do zamiany słów na cechy.

Przykład dla pliku `'small.csv'`:

W tym przykładzie i dalej pierwsze 5 wpisów są traktowane jako dane uczące, a ostatni jako dane testowe.

```
Dane wejściowe do funkcji (wczytane z pliku):
['just plain boring',
 'entirely predictable and lacks energy',
 'no surprises and very few laughs',
 'very powerful',
 'the most fun film of the summer']

Macierz cech:
[[0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0]
 [1 0 1 1 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0]
 [1 0 0 0 1 0 0 0 0 1 0 1 0 0 0 0 0 1 0 1]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1]
 [0 0 0 0 0 1 1 0 0 0 1 0 1 0 0 0 1 0 1 0]]

Słownik mapujący słowa na cechy:
{'and': 0, 'boring': 1, 'energy': 2, 'entirely': 3, 'few': 4, 'film': 5, 'fun': 6,
'just': 7, 'lacks': 8, 'laughs': 9, 'most': 10, 'no': 11, 'of': 12, 'plain': 13,
'powerful': 14, 'predictable': 15, 'summer': 16, 'surprises': 17, 'the': 18, 'very': 19}
```

Dla pliku `spam_prepared.csv`, macierz cech z danych uczących powinna mieć rozmiar (shape): `(5000, 8077)`, a słownik mapujący słowa na cechy powinien zawierać 8077 wartości.

4. Przygotuj funkcje `trainNB` zgodnie z wytycznymi z wykładu. Funkcja powinna przyjmować macierz cech X oraz odpowiadające jej etykiety y. Na wyjściu funkcja powinna zwrócić struktury *logprior* oraz *loglikelyhood* zawierające zlogarytmowane prawdopodobieństwa wystąpienia poszczególnych klas oraz poszczególnych słów w klasach. Funkcja powinna wykonywać następujące kroki:

   (a) Utwórz dwa słowniki: *logprior* oraz *loglikelyhood*.

   (b) Przygotuj zmienną $Xcount = X.sum() + X.shape[1]$ - jest to zmienna przedstawiająca sumę wystąpień wszystkich słów wraz z poprawką Laplace'a.

   (c) Dla każdej klasy c występującej w y wykonaj:

      i. $logprior[c] = log(\frac{\text{liczba dokumentów klasy } c \text{ w } y}{\text{łączna liczba dokumentów w } y}) = log(\frac{(y==c).sum()}{len(y)})$

      ii. $loglikelyhood[c] = log(\frac{\text{wystąpienia poszczególnych słów w klasie } c \, + \, 1}{Xcount}) = log(\frac{(X[y==c]).sum(axis=0)+1}{Xcount})$

      iii. Zwróć słowniki *logprior* i *loglikelyhood*

Podane powyżej kroki będą działać, jeżeli X i y zostały przygotowane poprawnie.

Przykład dla pliku `small.csv`:

```
Logprior:
{0: -0.5108256237659907, 1: -0.916290731874155}

Loglikelyhood (zaokrąglone do 4 miejsc po przycinku)
{
    0: array([-2.6391, -3.0445, -3.0445, -3.0445, -3.0445, -3.7377, -3.7377, -3.0445,
    -3.0445, -3.0445, -3.7377, -3.0445, -3.7377, -3.0445, -3.7377, -3.0445, -3.7377,
    -3.0445, -3.7377, -3.0445]),
```

```
    1: array([-3.7377, -3.7377, -3.7377, -3.7377, -3.7377, -3.0445, -3.0445, -3.7377,
    -3.7377, -3.7377, -3.0445, -3.7377, -3.0445, -3.7377, -3.0445, -3.7377, -3.0445,
    -3.7377, -3.0445, -3.0445])
}
```

5. Przygotuj funkcję $predictNB(text, logprior, loglikelihood, word\_to\_feature)$. Która zwróci do której klasy powinien zostać przydzielony $text$ na podstawie wyuczonych wcześniej prawdopodobieństw. Funkcja powinna wykonać następujące kroki:
   (a) Utworzyć słownik $prob\_sum$.
   (b) Dla każdej z klas w $logprior$ wykonaj:
      i. Ustaw prawdopodobieństwo a priori dla wyniku: $prob\_sum[c] = logprior[c]$
      ii. Dla każdego słowa w w $text$:
         A. Sprawdź czy w występuje w $word\_to\_feature$. Jeżeli nie to zignoruj to słowo.
         B. Jeżeli tak, to oblicz numer cechy i dodaj jego loglikelyhood do wyniku:
         $j = word\_to\_feature[w]$
         $prob\_sum[c] += loglikelyhood[c][j]$
   (c) Jako wynik zwróć tą z dwóch klas, która ma większą wartość $prob\_sum$

   Przykład dla pliku `small.csv`:

```
Dokument testowy:
predictable with no fun

Obliczone prawdopodobieństwa (zaokrąglone):
{0: -10.3375, 1: -11.4362}

Predykowana klasa: 0 (bo -10 > -11)
```

6. Uruchom funkcję $predictNB$ dla każdego dokumentu w zbiorze, tworząc listę (lub array) predykowanych klas 0 i 1. Następnie oblicz dokładność klasyfikacji na zbiorze testowym jako:

$$acc = \frac{\text{liczba poprawnie sklasyfikowanych dokumentów}}{\text{liczba wszystkich dokumentów}}$$

7. Przygotuj macierz konfuzji dla danych testowych. Macierz konfuzji odpowiada na pytanie, ile obiektów z każdej klasy zostało sklasyfikowanych poprawnie, a ile zostało pomylonych z inną klasą.

   Przykładowa macierz konfuzji i dokładność:

```
Dokładność: 93%

Macierz konfuzji:
True:      0     1
NB: 0:  1360   108
NB: 1:     0   104
```

   W powyższym przykładzie widać, że mimo tego, że mamy dosyć wysoką dokładność, tylko połowa spamu została rozpoznana poprawnie (104 z 212 dokumentów). Stąd przy klasyfikacji zawsze warto spojrzeć na macierz konfuzji. Uzyskaną dokładność i macierz konfuzji proszę wrzucić jako komentarz na końcu pliku.

Skala ocen

- Na ocenę 3.0: Zadania 1-3
- Na ocenę 4.0: Zadania 1-5
- Na ocenę 5.0: Zadania 1-7