Semestr zimowy 2021

# Czwarte laboratorium z przedmiotu Wprowadzenie do sztucznej inteligencji


***Autor***
Łukasz Jaremek
310710

***Data***
grudzień 2021

Praca wykonana samodzielnie.

# Użyte technologie

Język:
Python 3.8.8

Biblioteki ponad standardowe:
Brak

# Regresja i klasyfikacja

## Problem:

Implementacja drzewa decyzyjnego z algorytmem ID3, przeprowadzenie klasyfikacji motodą k-krotnej walidacji krzyżowej.

# Dane

Sprawdźmy na początku jak wyglądają nasze dane.
Jest to zbiór zawierający 12960 rekordów, każdy posiada 9 parametrów, z czego ostatni jest wartością poszukiwaną, którą będziemy chcieć wydedukować na podstawie pozostałych ośmiu.

Wartości kluczowe oraz ich ilość wystąpień:

| Nazwa | Ilość wystąpień |
|---|---|
| recommend | 2 |
| priority | 4266 |
| not_recom | 4320 |
| very_recom | 238 |
| spec_prior | 4044 |

Jak widać, dwie klasy (recommend, very_recom) są bardzo nieliczne, z czego liczba wystąpień pierwszej jest bliska zeru. Będzie to z pewnością miało wpływ na wyniki przeprowadzonych testów.

# Podział danych i klasyfikacja

Podział danych polega na podzieleniu całego zbioru na X równych (o ile to możliwe) części. Pierwsze X-1 części (treningowych) posłuży do trenowania drzew losowych, natomiast ostatni zbiór (testowy) posłuży do sprawdzenia, czy las losowy poprawnie klasyfikuje dane (decyzja lasu losowego to najczęściej występujący wynik we wszystkich drzewach).

Zacznijmy od podzielenia zbioru na 3 części. Otrzymujemy wtedy wynik:

| Otrzymana → Oczekiwana ↓ | not_recom | priority | recommend | spec_prior | very_recom |
|---|---|---|---|---|---|
| not_recom | 1449 | 0 | 0 | 0 | 0 |
| priority | 433 | 886 | 0 | 32 | 34 |
| recommend | 0 | 0 | 0 | 0 | 0 |
| spec_prior | 293 | 48 | 0 | 1039 | 0 |
| very_recom | 27 | 33 | 1 | 0 | 45 |

Średnie Acc dla 5 prób: 0.82

Podział na 5 części:

| Otrzymana → Oczekiwana ↓ | not_recom | priority | recommend | spec_prior | very_recom |
|---|---|---|---|---|---|
| not_recom | 837 | 0 | 0 | 0 | 0 |
| priority | 162 | 646 | 0 | 38 | 14 |
| recommend | 0 | 0 | 0 | 0 | 0 |
| spec_prior | 99 | 36 | 0 | 696 | 0 |
| very_recom | 8 | 25 | 0 | 0 | 31 |

Średnie Acc dla 5 prób: 0.92

Możemy zauważyć, że po przekątnej zawsze jest największa liczba w wierszu/kolumnie, co oznacza, że las w większości przypadków poprawnie typuje wynik.
Natomiast w kolumnie i wierszu z klasą recommend jest zawsze liczba bliska zeru – wynika to z bardzo niskiej liczebności tej klasy (dwa wystąpienia). Przy podziale danych na więcej niż 4 części, dochodzi do sytuacji, gdzie dwa drzewa (które dostały do treningu wiersz z tą klasą) podadzą poprawny wynik, natomiast reszta (3 lub więcej) nie zna takiej klasy, więc poda wynik, który bazuje na innych kolumnach, co będzie zwykłym zgadywaniem i nie gwarantuje poprawności.

# Macierze pomyłek

Sprawdźmy jak wyglądaja macierze pomyłek. Sprawdziny to na skrajnej klasie (**recommend** z liczbą wystąpień rowną 2) oraz na przeciętnej (**priority**).

**Recommend** dla k=5:

| | Wartości rzeczywiste | |
|---|---|---|
| Wartości oczekiwane | TP = 0 | FP = 0 |
| | FN = 0 | TN = 2317 |

Widzimy, że TN został obliczony poprawnie, natomiast nie widać ani jednego wystąpienia dla FP, FN czy TP, co wynika z tego, że prawdopodobieństwo trafienia na klasę recommend jest bardzo niskie.

**Priority** dla k=5:

| | Wartości rzeczywiste | |
|---|---|---|
| Wartości oczekiwane | TP = 667 | FP = 82 |
| | FN = 153 | TN = 1640 |

Tutaj natomiast widać bardziej poprawny rozkład, TP oraz TN wyraźnie przeważają nad FN i FP co świadczy o poprawnym działaniu lasu.

Wskaźniki dodatkowe (uwzględniające wszystkie klasy) dla powyższych danych:

| Nazwa | Wartość |
|---|---|
| TPR | 0.86 |
| FPR | 0.04 |
| PPV | 0.90 |
| Acc | 0.91 |

# Wpływ podziału danych na wyniki

Sprawdźmy jak parametr k (ilość zbiorów, na które dzielimy zbiór pierwotny) wpłynie na wyniki.

Dla 5 prób:

| k | Średnie Acc na zbiorze testowym | Średnie Acc na zbiorze treningowym |
|---|---|---|
| 3 | 0.82 | 0.95 |
| 5 | 0.92 | 0.96 |
| 8 | 0.950 | 0.93 |
| 13 | 0.952 | 0.97 |
| 30 | 1.0 | 1.0 |

Jak widać, zwiększanie **k** wpływa na wzrost **Acc** na zbiorze testowym jak i treningowym, co świadczy o tym, że zbiory są do siebie bardzo podobne oraz, że nie może dojść w tym przypadku do przetrenowania na takim zbiorze testowym. Dane są dzielone w sposób pseudo - losowy co świadczy o równomiernym rozkładzie danych w zbiorze pierwotnym.

# Przykładowe drzewo

Można powiedziec, że drzewo decyzyjne jest bytem reprezentowalnym (da się przestawić w zrozumiały dla człowieka sposób) – co może być prawdą, lecz nie musi.
Drzewo dla zbioru danych z kilkoma / kilkunastoma wszystkimi klasami da się narysować na kartce A4, natomiast drzewo stworzone z zadanego zbioru danych jest tak duże, że aż nieczytelne. Wyobraźmy sobie teraz las, złożony z takich potężnych drzew – bez wielogodzinnej analizy nic byśmy z tego nie wywnioskowali.

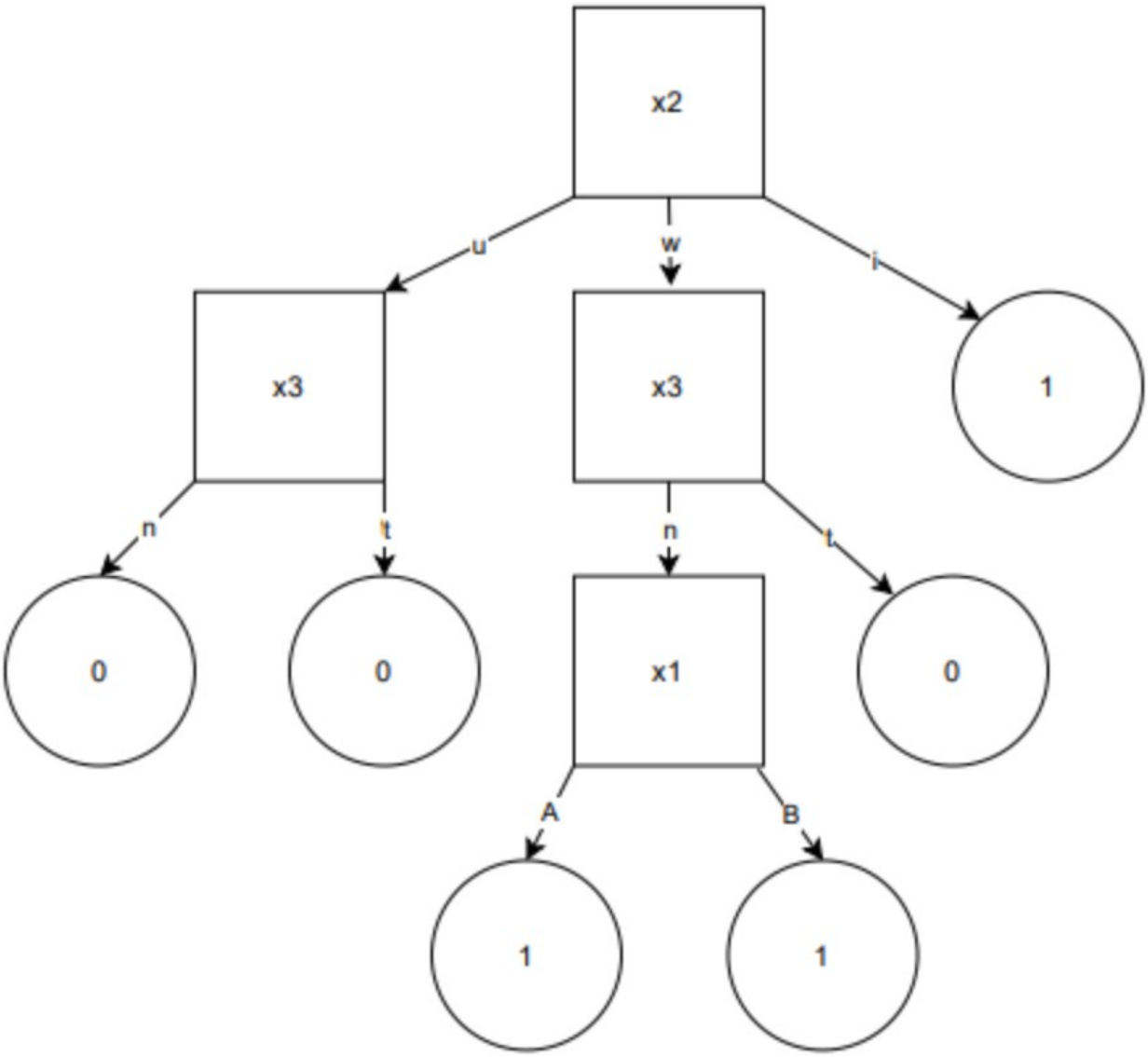
Dlatego przykładowe drzewo, przedstawione niżej powstało z sztucznie wygenerowane zbioru danych w postaci:

| x1 | x2 | x3 | y |
|---|---|---|---|
| A | w | t | 0 |
| A | w | n | 1 |
| B | u | t | 0 |
| B | i | t | 1 |
| A | u | n | 0 |
| B | w | n | 1 |

Drzewo (rysowane w Pythonie):

```
NULL -> x2
        u -> x3
                n -> 0
                t -> 0
        w -> x3
                t -> 0
                n -> x1
                        A -> 1
                        B -> 1
        i -> 1
```

To samo drzewo w bardziej czytelnej postaci:



# Podsumowanie laboratorium

Drzewo decyzyjne to względnie prosty, lecz co najważniejsze wytłumaczalny algorytm na szacowanie wyniku na podstawie danych. Nie wymaga on znajomości zaawansowanej mateamtyki, a na raz napisanym drzewie można klasyfikować wiele zbiorów danych, o ile spełniają pewne kryteria. Skuteczność drzewa sięga 90% co przy tak małym zaangażowaniu w jego implementację nie jest złym wynikiem.

Isotną wadą natomiast jest nieradzenie sobie z nowymi danymi, których nie było w zbiorze treningowym, lub było ich mało. Lub podejmowanie decyzji w momentach, gdy dwie potencjalne odpowiedzi mają taką sama szansę wystąpienia.