Wstęp do sztucznej inteligencji

Lab_04 – Klasyfikacja za pomocą algorytmu ID3

Bartosz Latosek

310 790

1. Wstęp

Celem ćwiczenia jest budowa i analiza algorytmu ID3 za pomocą którego dokonamy klasyfikacji zadanego zbioru danych.

Użyte biblioteki:

- *Pygame*
- *Argparse*
- *MatPlotLib*

1.1 Opis badanego zbioru danych

Badany zbiór danych składa się z 8 – atrybutowych obiektów będących reprezentacją ucznia przedszkola. Klasą ucznia jest prawdopodobieństwo przyjęcia go do placówki na rok szkolny.

| class values

not_recom, recommend, very_recom, priority, spec_prior

| attributes

parents: usual, pretentious, great_pret.
has_nurs: proper, less_proper, improper, critical, very_crit.
form: complete, completed, incomplete, foster.
children: 1, 2, 3, more.
housing: convenient, less_conv, critical.
finance: convenient, inconv.
social: nonprob, slightly_prob, problematic.
health: recommended, priority, not_recom.

Obiekty zapisywane są w pliku tekstowym w formacie *csv* (bez nagłówka) ***** jako lista atrybutów zakończona klasą, do której obiekt przynależy.

Przykładowe dane:

usual,proper,complete,1,convenient,convenient,nonprob,recommended,recommend

pretentious,very_crit,completed,1,less_conv,inconv,problematic,priority,spec_prior

pretentious,very_crit,incomplete,1,less_conv,convenient,problematic,recommended,spec_prior

---

***** W tym momencie zaznaczę, że moja implementacja algorytmu nie jest zgeneralizowana tylko do zbioru danych *nursery*, ale do dowolnie tak zapisanego zbioru danych.

Rozkład wartości atrybutów:

```
{'usual': 4320, 'pretentious': 4320, 'great_pret': 4320}
{'proper': 2592, 'less_proper': 2592, 'improper': 2592, 'critical': 2592, 'very_crit': 2592}
{'complete': 3240, 'completed': 3240, 'incomplete': 3240, 'foster': 3240}
{'1': 3240, '2': 3240, '3': 3240, 'more': 3240}
{'convenient': 4320, 'less_conv': 4320, 'critical': 4320}
{'convenient': 6480, 'inconv': 6480}
{'nonprob': 4320, 'slightly_prob': 4320, 'problematic': 4320}
{'recommended': 4320, 'priority': 4320, 'not_recom': 4320}
```

Z powyższej grafiki wynika, że zbiór danych ułożony jest tak, aby każdy atrybut występował dokładnie tyle samo razy w całym zbiorze.

Rozkład wartości atrybutów ze względu na klasy:

Aby w pełni pokazać analizę zależności wartości atrybutu od klasy obiektu należałoby w sprawozdaniu zamieścić **7** (*liczba atrybutów)* *** 5** (*liczba klas*) wykresów zależności. W związku z czym jako dowód przeanalizowania tej części zadania zamieszczam poniższą tekstową reprezentację tej zależności wraz z wyciągniętymi z niej wnioskami.

```
recommend
{'usual': 2}
{'proper': 2}
{'complete': 2}
{'1': 2}
{'convenient': 2}
{'convenient': 2}
{'nonprob': 1, 'slightly_prob': 1}
{'recommended': 2}


spec_prior
{'usual': 758, 'pretentious': 1264, 'great_pret': 2022}
{'critical': 1264, 'very_crit': 1518, 'improper': 758, 'proper': 252, 'less_proper': 252}
{'complete': 888, 'completed': 968, 'incomplete': 1052, 'foster': 1136}
{'2': 968, '3': 1136, 'more': 1136, '1': 804}
{'critical': 1608, 'convenient': 1052, 'less_conv': 1384}
{'convenient': 1856, 'inconv': 2188}
{'nonprob': 1200, 'slightly_prob': 1200, 'problematic': 1644}
{'priority': 2466, 'recommended': 1578}


not_recom
{'usual': 1440, 'pretentious': 1440, 'great_pret': 1440}
{'proper': 864, 'less_proper': 864, 'improper': 864, 'critical': 864, 'very_crit': 864}
{'complete': 1080, 'completed': 1080, 'incomplete': 1080, 'foster': 1080}
{'1': 1080, '2': 1080, '3': 1080, 'more': 1080}
{'convenient': 1440, 'less_conv': 1440, 'critical': 1440}
{'convenient': 2160, 'inconv': 2160}
{'nonprob': 1440, 'slightly_prob': 1440, 'problematic': 1440}
{'not_recom': 4320}


priority
{'usual': 1924, 'pretentious': 1484, 'great_pret': 858}
{'proper': 1344, 'less_proper': 1344, 'improper': 904, 'critical': 464, 'very_crit': 210}
{'complete': 1152, 'completed': 1092, 'incomplete': 1038, 'foster': 984}
{'1': 1206, '2': 1092, '3': 984, 'more': 984}
{'convenient': 1618, 'less_conv': 1396, 'critical': 1252}
```

```
{'convenient': 2244, 'inconv': 2022}
{'nonprob': 1515, 'slightly_prob': 1515, 'problematic': 1236}
{'priority': 1854, 'recommended': 2412}


very_recom
{'usual': 196, 'pretentious': 132}
{'proper': 130, 'less_proper': 132, 'improper': 66}
{'complete': 118, 'completed': 100, 'incomplete': 70, 'foster': 40}
{'1': 148, '2': 100, '3': 40, 'more': 40}
{'convenient': 208, 'less_conv': 100, 'critical': 20}
{'inconv': 110, 'convenient': 218}
{'nonprob': 164, 'slightly_prob': 164}
{'recommended': 328}
```

Najciekawszy wydaje się przypadek klasy *recommend,* który można praktycznie od razu skategoryzować ze względu na jednoznaczność aż 7 z 8 atrybutów.

Drugim ciekawym przypadkiem jest klasa *not_recom* wraz z ostatnim parametrem *health.* Możemy tutaj zauważyć, że wartość *not_recom* jednoznacznie definiuje klasę *not_recom*. (W żadnej innej klasie nie występuje ta wartość tego konkretnego atrybutu).

Możemy też zauważyć, że ten sam atrybut dla klasy *very_recom* nie przyjmuje innej wartości niż *recommended* więc można by było na tej podstawie jednoznacznie wykluczać obiekty.

Można by jeszcze na siłę znaleźć kilka mniej oczywistych zależności, ale nie są one tak widoczne i znaczące jak powyższe trzy, w związku z czym pominę je w dalszej analizie problemu.

1.2 Algorytm

data: *dataset, attributes*

root = *DecisionTreeNode*

if $\nexists\, \boldsymbol{c} \in dataset$, $\dot{\boldsymbol{z}}\boldsymbol{e}\ \boldsymbol{c} \rightarrow \boldsymbol{class} != \boldsymbol{C}$ **:**
  root->isLeaf = **True**
  root->classValue = **C**
else if !attribures:
  root->isLeaf = **True**
  root->classValue = *most common Class in dataset*
else:
  H = *GeneralEntropy(dataset)*
  all_entropies = *EntropiesForAttributes(dataset, attrubutes)*
  inf_gain = **[**H – entropy **for** entropy in all_entropies**]**

  root->decidingAttribute = *Attribute with highest inf_gain*
  for each value **of** *Attribute with highest inf_gain:*
    new_dataset = dataset **-** [data **in** dataset **with** data->Attribute with highest inf_gain **!=** value]
    new_node = **id3(***new_dataset, attributes – Attribute with highest inf_gain***)**
    new_node->value = *value*
    root->children **+=** *new_node*

return root

Gdzie:

dataset – badany zbiór danych (początkowo wszystkie dane)

attributes – indeksy atrybutów branych pod uwagę w obecnej iteracji

1.3 Przykładowe drzewo

Utworzenie drzewa dla zbioru danych *nursery*.txt mija się z celem, gdyż jego rozłożystość wpływa negatywnie na czytelność schematu. W związku z czym poniższa wizualizacja przedstawia wynik działania algorytmu ID3 dla zbioru danych: (czy dzień nadaje się do gry w tenisa)

OUTLOOK ,TEMP,HUMIDITY,WIND,DECISION

sunny,hot,high,weak,no

sunny,hot,high,strong,no

overcast,hot,high,weak,yes

rain,mild,high,weak,yes

rain,cool,normal,weak,yes

rain,cool,normal,strong,no

overcast,cool,normal,strong,yes

sunny,mild,high,weak,no

sunny,cool,normal,weak,yes

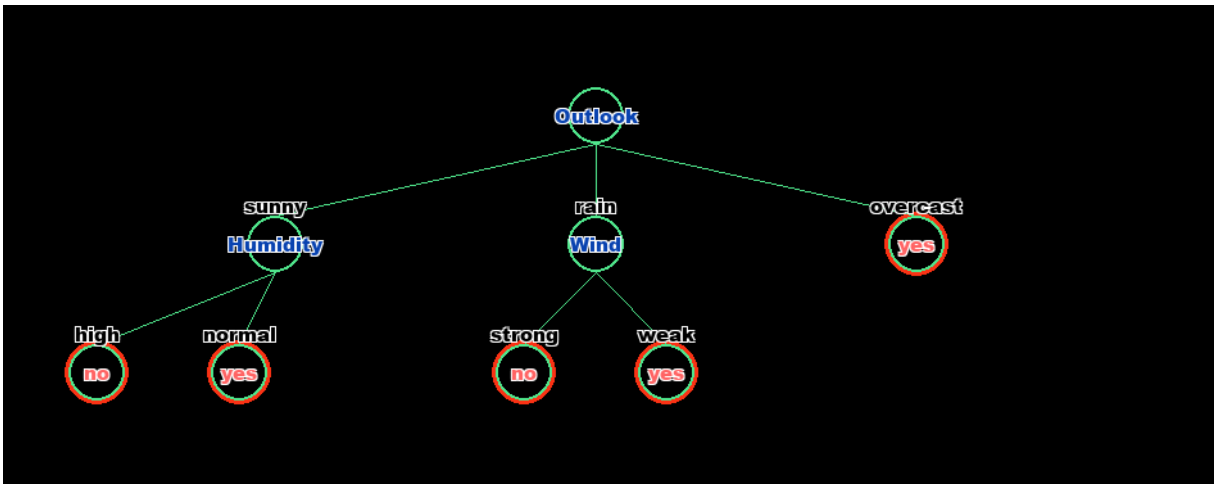
rain,mild,normal,weak,yes

sunny,mild,normal,strong,yes

overcast,mild,high,strong,yes

overcast,hot,normal,weak,yes

rain,mild,high,strong,no



Drzewo wygenerowane za pomocą klasy Visualize

2. Analiza działania algorytmu

2.1 Analiza działania algorytmu w zależności od miejsca podziału zbioru danych na zbiór testowy i uczący

I Zbiór danych losowy **I**

Stosunek wielkości Z.U. do Z.T	**Stosunek poprawnych do niepoprawnych dedukcji klasy**	**Skuteczność**
12960 : 0 (Z.U = Z.T)	12960 : 0	100,0%
9720 : 3240	3205 : 35	98,9%
6480 : 6480	6280 : 200	96,9%
3240 : 9720	9147 : 573	94,1%
960 : 12000	10826 : 1174	90,2%
100 : 12860	10546 : 2314	82,0%
10 : 12980	8576 : 4373	66,1%

Kolejność danych w ogólnym zbiorze danych była każdorazowo losowana a stosunek ustalany poprzez dzielenie tego zbioru na 2 podzbiory zgodne ze stosunkiem podanym w 1 kolumnie tabeli. Co ciekawe nawet przy bardzo małym zbiorze uczącym (ostatni wiersz tabeli) , drzewo jest w stanie poprawnie przewidzieć klasę w 2/3 przypadków.

I Zbiór danych wstępnie posortowany **I**

Stosunek wielkości Z.U. do Z.T	**Stosunek poprawnych do niepoprawnych dedukcji klasy**	**Skuteczność**
12960 : 0 (Z.U = Z.T)	12960 : 0	100.0%
9720 : 3240	2908 : 332	89.1%
6480 : 6480	4728 : 1752	72.9%
3240 : 9720	5932 : 3788	61.0%
960 : 12000	7292 : 4708	60,7%
100 : 12860	7416 : 5444	57,7%
10 : 12980	6839 : 6111	52.8%

W przypadku posortowanych danych wyniki są znacznie gorsze niż te dla losowych. Posortowane dane sprawiają, że w przypadkach zawartych w dalszych wierszach tabeli zbiór testowy zawiera klasy nie znajdujące się w zbiorze uczącym.

==========

Wnioski:

Z tabeli wynika, że jesteśmy w stanie osiągnąć wysoką skuteczność modelu przy pomocy stosunkowo małego zbioru uczącego. Pierwsze 3 wiersze tabeli losowej i pierwszy wiersz wstępnie posortowanej można uznać za przypadek przeuczenia. Przypadek niedouczenia nie wystąpił w pierwszej, a w ostatniej wystąpił przy bardzo małym zbiorze uczącym co było przewidywalne.

1.2 Macierz błędów

Macierz błędów będzie wyznaczana na podstawie pomieszanego zbioru danych, na zbiorze danych testowym będącym w stosunku 1 : 2 ze zbiorem uczącym.

	Very_recom	Spec_prior	Priority	Not_recom	Recommend
Very_recom	71	0	14	0	0
Spec_prior	0	986	15	0	0
Priority	10	11	1066	0	0
Not_recom	0	0	0	1065	0
Recommend	2	0	0	0	0

macierz błędów

	Very_recom	Spec_prior	Priority	Not_recom	Recommend
Recall	85.5%	98.9%	97.3%	100%	0
Fallout	0.44%	0.66%	0.98%	0%	<0.01%
Precision	83.5%	98.5%	98%	100%	0%
Accuracy	99.2%	99.2%	98.5%	100%	99.9%
F1-score	84.5%	98.7%	97.7%	100%	0%

Wartości metryk

RECALL	FALLOUT	PRECISION	ACCURACY	F1-SCORE
98.3%	0.04%	98.3%	99.3%	98.3%

Uogólnione wartości metryk dla macierzy

Z powyższej tabeli widać, że algorytm działa zgodnie z oczekiwaniami i w większej ilości przypadków poprawnie wydedukuje klasę podanego obiektu.

1.3 Walidacja krzyżowa w zależności od parametru k

W przypadku analizy walidacji krzyżowej porównywane będą metryki uogólnione dla macierzy błędów dla każdego zbioru testowego, wydzielonego z pomieszanego zbioru danych.

k = 2

1.

	Not_recom	Priority	Recommend	Spec_prior	Very_recom
Not_recom	2168	0	0	0	0
Priority	0	2047	0	55	40
Recommend	0	0	0	0	2
Spec_prior	0	40	0	1967	0
Very_recom	0	32	0	0	129

Rec: 97.3% Fall: 0.06% Prec: 97.4% Acc: 98.9% F1: 97.3%

2.

	Not_recom	Priority	Recommend	Spec_prior	Very_recom
Not_recom	2152	0	0	0	0
Priority	0	2040	0	48	36
Recommend	0	0	0	0	0
Spec_prior	0	78	0	1959	0
Very_recom	0	55	0	0	112

Rec: 96.6% Fall: 0.08% Prec: 96.6% Acc: 98.6% F1: 96.6%

k = 3

1.

	Not_recom	Priority	Recommend	Spec_prior	Very_recom
Not_recom	1426	0	0	0	0
Priority	0	1397	0	34	10
Recommend	0	0	0	0	1
Spec_prior	0	16	0	1313	0
Very_recom	0	28	0	0	95

Rec: 97.94% Fall: 0.52% Prec: 97.94% Acc: 99.18% F1: 97.94%

2.

	Not_recom	Priority	Recommend	Spec_prior	Very_recom
Not_recom	1436	0	0	0	0
Priority	0	1379	0	17	11
Recommend	0	0	0	0	0
Spec_prior	0	29	0	1355	0
Very_recom	0	20	0	0	73

Rec: 98.22% Fall: 0.45% Prec: 98.22% Acc: 99.29% F1: 98.22%

3.

	Not_recom	Priority	Recommend	Spec_prior	Very_recom
Not_recom	1458	0	0	0	0
Priority	0	1360	0	26	32
Recommend	0	0	0	0	1
Spec_prior	0	18	0	1313	0
Very_recom	0	25	0	0	87

Rec: 97.64% Fall: 0.59% Prec: 97.64% Acc: 99.06% F1: 97.64%

W związku z tym, że wartości metryk jak i dane w macierzy błędów nie różnią się znacząco, w kolejnych przypadkach k będę wypisywać dane jedynie dla jednego ze zbiorów testowych.

k = 5

1.

	Not_recom	Priority	Recommend	Spec_prior	Very_recom
Not_recom	855	0	0	0	0
Priority	0	826	0	6	1
Recommend	0	0	0	0	0
Spec_prior	0	12	0	823	0
Very_recom	0	16	1	0	52

Rec: 98.61% Fall: 0.35% Prec: 98.61% Acc: 99.44% F1: 98.61%

k = 7

1.

	Not_recom	Priority	Recommend	Spec_prior	Very_recom
Not_recom	605	0	0	0	0
Priority	0	612	0	0	10
Recommend	0	0	0	0	1
Spec_prior	0	11	0	562	0
Very_recom	0	8	1	0	35

Rec: 98.0% Fall: 0.5% Prec: 98.0% Acc: 99.2% F1: 98.0%

k = 10

1.

	Not_recom	Priority	Recommend	Spec_prior	Very_recom
Not_recom	416	0	0	0	0
Priority	0	408	0	0	7
Recommend	0	0	0	0	1
Spec_prior	0	6	0	423	0
Very_recom	0	1	0	0	34

Rec: 98.84% Fall: 0.29% Prec: 98.84% Acc: 99.54% F1: 98.84%

k = 20

1.

	Not_recom	Priority	Recommend	Spec_prior	Very_recom
Not_recom	214	0	0	0	0
Priority	0	201	0	0	4
Recommend	0	0	0	0	0
Spec_prior	0	2	0	213	0
Very_recom	0	0	0	0	14

Rec: 99.07% Fall: 0.23% Prec: 99.07% Acc: 99.63% F1: 99.07%

Wniosek

Jak widać, wartość parametru k przy losowo poukładanych danych wejściowych i walidacji krzyżowej nie ma znaczenia jeżeli chodzi o wartości metryk macierzy błędów. Jest to spowodowane tym, że przez losowanie dane są porozkładane równomiernie pomiędzy wszystkie zbiory uczące i zbiór testowy.