Wydział Elektroniki
i Technik Informacyjnych
POLITECHNIKA WARSZAWSKA

Wstęp do Sztucznej Inteligencji

Ćwiczenie nr 4:
Regresja i klasyfikacja

Kaczmarski Robert
293377

Warszawa 2021

Polecenie

Zadanie polega klasyfikacji przyjęcia dzieci do przedszkola na podstawie informacji o strukturze i finansach rodziny. Należy dokonać implementacji drzewa decyzyjnego przy pomocy algorytmu ID3 i przeprowadzić klasyfikację metodą k-krotnej walidacji krzyżowej dla zadanego zbioru danych.

Użyte narzędzia

- pandas==1.3.5
- numpy==1.21.4
- matplotlib==3.5.1

Wyniki

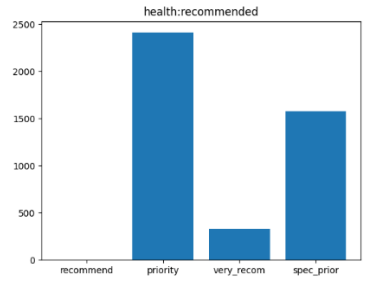
Badany zbiór danych zawiera 12960 obserwacji, 8 atrybutów, 5 klas, których rozkład jest opisany w Tabeli 1. Trzy klasy mają podobną liczność ok. 4000, a dwie mają małą liczność

Tabela 1. Rozkład klas (liczba instancji na klasę).

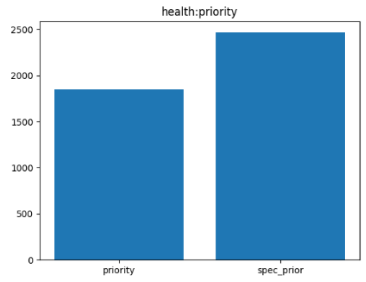
class	N	N[%]
not_recom	4320	33,333%
recommend	2	0,015%
very_recom	328	2,531%
priority	4266	32,917%
spec_prior	4044	31,204%

Analiza

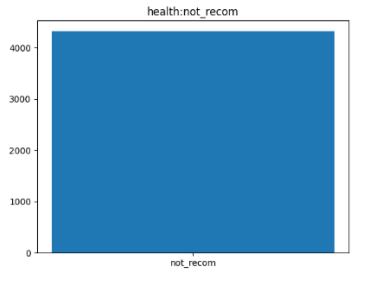
Na początku przeprowadzona została wstępna analiza zbioru danych dla wszystkich wartości każdego z atrybutów. Analiza wykazała, że najbardziej znaczącym czynnikiem mającym wpływ na decyzję przyjęcia dziecka do przedszkola jest stan zdrowia co przedstawiają Rys. 1-3.



Rys. 1. Wyniki klasyfikacji dla wartości "recommended" atrybutu "health".



Rys. 2. Wyniki klasyfikacji dla wartości "priority" atrybutu "health".



Rys. 3. Wyniki klasyfikacji dla wartości "not_recom" atrybutu "health".

Zły ogólny stan zdrowia rodziny automatycznie dyskwalifikuje dziecko na starcie. Inne zauważone zależności to np. im gorsze warunki domowe i finansowe tym rzadziej dziecko jest przyjmowane.

Wynika z tego, że przedszkola preferują dzieci ze zdrowej, majętnej, bezproblemowej rodziny, co jest jak najbardziej zrozumiałe.

Podział danych

Aby sprawdzić wpływ podziału danych na zbiór trenujący i testowy przeprowadzone zostały eksperymenty zmieniając proporcje podziału co 10%. Drzewo decyzyjne zostało stworzone na podstawie zbioru trenującego, a następnie przeprowadzona została klasyfikacja zbioru testowego. Wyniki przedstawia Tabela 2.

Tabela 2. Badanie podziału zbioru na skuteczność modelu.

Proporcje zbiorów trenujący / testowy	Procent poprawnie zaklasyfikowanych dzieci danej klasy				
	not_recom	recommended	very_recom	priority	spec_prior
0,1 / 0,9	100%	0%	37,87%	87,27%	88,25%
0,2 / 0,8	100%	0%	56,98%	87,96%	87,48%
0,3 / 0,7	100%	0%	56,77%	90,16%	92,71%
0,4 / 0,6	100%	0%	61,86%	91,63%	94,86%
0,5 / 0,5	100%	0%	68,32%	92,84%	96,37%
0,6 / 0,4	100%	0%	73,81%	94,26%	96,62%
0,7 / 0,3	100%	0%	73,40%	96,06%	97,50%
0,8 / 0,2	100%	0%	77,78%	95,80%	97,95%
0,9 / 0,1	100%	-	82,76%	97,22%	98,52%

Pierwsze co rzuca się w oczy to perfekcyjne dopasowanie *not_recom* i zerowe *recommended*. Dzieje się tak, ponieważ algorytm ID3 najpierw dzieli zbiór ze względu na atrybut *health*, a jak już wcześniej zostało powiedziane, wszystkie dane z klasą not_recom mają w kolumnie *health* wartość *not_recom*. Oznacza to, że na pierwszym poziomie drzewa występuje liść z klasą *not_recom*, dlatego łatwo jest takie dziecko zaklasyfikować. Klasa *recommended* występuje tylko dla dwóch przypadków, dlatego może być trudna do zaklasyfikowania. Poza tymi spostrzeżeniami widać wyraźnie, że dla pozostałych klas zwiększenie proporcji zbioru do trenowania zwiększa liczbę poprawnie zaklasyfikowanych przyjęć dzieci do przedszkola.

(Gdy zbiór danych do trenowania posłużył zarówno do stworzenia modelu jak i przeprowadzenia testów, wszystkie klasy miały 100% dopasowania. Może to być oznaką przeuczenia modelu, ale nie mam pewności. Możliwe, że to wina jakiegoś buga w kodzie.)

K-krotna walidacja krzyżowa

Wejściowy zbiór danych dzielony jest na k podzbiorów o stałej liczności. K-1 podzbiorów stanowi zbiór do trenowania, a jeden podzbiór to zbiór testowy. Dla przypadku k = 1, podzbiorów stanowiących zbiór do trenowania będzie 1 - 1 = 0, czyli zbiór do trenowania jest pusty. Oznacza to, że zaimplementowany algorytm nie utworzy drzewa, więc nie ma sensu sprawdzać wyników dla tego przypadku. Zbadany zostanie wpływ parametru k na jakość klasyfikacji dla $k \in \{3,5,7,10,20\}$. Wyniki zostały zestawiony w Tabeli 3-7. Przed podziałem na podzbiory, wiersze w zbiorze danych zostały przetasowane z parametrem *random_state = 10*.

Tabela 3. Procent poprawnie zaklasyfikowanych dzieci danej klasy dla k=3.

K = 3	Minimum	Maksimum	Średnia	Odchylenie
not_recom	100%	100%	100%	0.00%
recommended	0%	0%	0%	0%
very_recom	61,11%	77,06%	68,88%	42,50%
priority	95,57%	95,99%	95,73%	0,03%
spec_prior	97,01%	97,42%	97,28%	0,04%

Tabela 4. Procent poprawnie zaklasyfikowanych dzieci danej klasy dla k=5.

K = 5	Minimum	Maksimum	Średnia	Odchylenie
not_recom	100%	100%	100%	0.00%
recommended	0%	0%	0%	0%
very_recom	65,28%	79,31%	72,77%	26,69%
priority	95,60%	97,14%	96,33%	0,31%
spec_prior	96,73%	98,33%	97,64%	0,31%

Tabela 5. Procent poprawnie zaklasyfikowanych dzieci danej klasy dla k=7..

K = 7	Minimum	Maksimum	Średnia	Odchylenie
not_recom	100%	100%	100%	0.00%
recommended	0%	0%	0%	0%
very_recom	58,33%	87,50%	73,92%	68,42%
priority	95,21%	98,39%	96,95%	0,97%
spec_prior	97,33%	98,75%	98,17%	0,33%

Tabela 6. Procent poprawnie zaklasyfikowanych dzieci danej klasy dla k=10..

K = 10	Minimum	Maksimum	Średnia	Odchylenie
not_recom	100%	100%	100%	0.00%
recommended	0%	0%	0%	0%
very_recom	47,96%	86,11%	74,58%	106,06%
priority	94,43%	98,77%	97,09%	1,37%
spec_prior	97,96%	99,28%	98,66%	0,19%

Tabela 7. Procent poprawnie zaklasyfikowanych dzieci danej klasy dla k=20..

K = 20	Minimum	Maksimum	Średnia	Odchylenie
not_recom	100%	100%	100%	0.00%
recommended	0%	0%	0%	0%
very_recom	40%	94,74%	78,38%	148,68%
priority	95,24%	99,51%	97,65%	1,64%
spec_prior	97,04%	100%	98,88%	0,67%

K-krotna walidacja krzyżowa nie wpłynęła na skuteczność klasyfikacji klas *not_recom* oraz *recommended*. Dla pozostałych klas, a mianowicie *very_recom*, *priority*, *spec_prior*, zwiększenie parametru k powoduje większy średni procent poprawnie zakwalifikowanych dzieci. Im większy

parametr k tym mniejszą wartość minimum udało się uzyskać w klasie *very_recom*. Również dla tej klasy duża wartość wariancji budzi pewne wątpliwości co do poprawności obliczeń.

Walidacji krzyżowej używa się m.in. w celu uniknięcia przeuczenia modelu. W zaimplementowanym algorytmie nie zaobserwowano przeuczenia, więc pod tym kątem k-krotna walidacja krzyżowa nie jest zbyt potrzebna.

Posortowany zbiór trenujący

Ściągnięty zbiór danych był już wstępnie posortowany, dlatego zbiór treningowy będzie posortowany względem indeksu wiersza. Porównana zostanie skuteczność modelu dla zbioru danych wstępnie posortowanych i dla zbioru, którego wiersze zostaną przetasowane. Przetasowanie danych odbywało się podczas podziału na zbiór treningowy i testowy poprzez ustawienie parametru random_state. Zbiór danych był podzielony w proporcji 9:1 na korzyść zbioru trenującego

Tabela 8. Porównanie efektywności modelu dla wstępnie posortowanego zbioru trenującego ze specjalnie pomieszanymi.

	Procent poprawnie zaklasyfikowanych dzieci danej klasy				
	not_recom	recommended	very_recom	priority	spec_prior
posortowane	100%	-	82,76%	97,22%	98,52%
random_state =2	100%	-	80,65%	96,91%	99,29%
random_state= 3	100%	-	78,38%	98,04%	99,30%
random_state =4	100%	-	84,62%	96,66%	98,01%
random_state =5	100%	-	76,32%	99,00%	97,57%

W zaimplementowanym drzewie decyzyjnym nie widać zależności między wstępnie posortowanym, a specjalnie pomieszanym zbiorze trenującym

Macierz pomyłek

Macierz pomyłek razem z miarami takimi jak precyzja, czułość, dokładność modelu została wyznaczona dla podziału zbioru danych na trenujący i testowy w stosunku 9:1 oraz dla wartości parametru random_state = 1.

Tabela 9. Macierz pomyłek.

		Klasa rzeczywista				
		not_recom	recommended	very_recom	priority	spec_prior
Klasa przewidywana	not_recom	429	0	0	0	0
	recommended	0	0	0	0	0
	very_recom	0	0	24	0	0
	priority	0	0	1	420	3
	spec_prior	0	0	0	3	400

W zaimplementowanym algorytmie ID3, gdy wartość jakiegoś atrybutu ze zbioru testowego nie pojawia się w zbiorze trenującym to taka obserwacja jest ignorowana, a dokładniej odpowiedzi przypisywany jest pusty string predykcji. Przy tworzeniu macierzy pomyłek w Tabeli 9. Pojawiły się łącznie 23 złe zaklasyfikowania, z czego 16 z nich to przypadek brakującego atrybutu. Te 16 przypadków nie będzie branych pod uwagę przy dalszym wyznaczaniu miar, co może wpłynąć na zawyżenie wyniku.

$$Precyzja_{not\_recom} = \frac{429}{429} = 100\%$$

$$Precyzja_{recommended} = \frac{0}{0} = ?$$

$$Precyzja_{very\_recom} = \frac{24}{24+1} = 96\%$$

$$Precyzja_{priority} = \frac{420}{420+3} = 99{,}29\%$$

$$Precyzja_{spec\_prior} = \frac{400}{400+3} = 99{,}26\%$$

$$Czułość_{not\_recom} = \frac{429}{429} = 100\%$$

$$Czułość_{recommended} = \frac{0}{0} = ?$$

$$Czułość_{very\_recom} = \frac{24}{24} = 100\%$$

$$Czułość_{priority} = \frac{420}{420+1+3} = 99{,}06\%$$

$$Czułość_{spec\_prior} = \frac{400}{400+3} = 99{,}26\%$$

$$Dokładność = \frac{429+0+24+420+400}{429+0+24+420+400+1+3+3} = 99{,}45\%$$

$$Dokładność_{uwzględniająca\_16\_braków} = \frac{429+0+24+420+400}{429+0+24+420+400+1+3+3+16} = 98{,}21\%$$

Wyniki oceny jakości klasyfikacji są bardzo wysokie. Jedynie klasa *recommended*, której dane stanowią 0,015% całego zbioru, nie daje się zaklasyfikować. To bardzo dziwne, ponieważ wcześniej (kilka dni temu), podczas przeprowadzonych testów sprawdzających poprawność działania tworzenia drzewa decyzyjnego, udawało się poprawnie zaklasyfikować dzieci do klasy *recommended*. Takie podejrzane zachowanie może świadczyć o błędzie w kodzie.

Przykładowy wygląd drzewa dla zaimplementowanego algorytmu

Z racji tego, że zbiór danych ma 8 atrybutów i każdy z nich ma kilka wartości to pokazanie pełnego drzewa jest niebotycznie trudnym zadaniem, dlatego poniżej zamieszczam przykładowe drzewo dla zbioru danych zawierających pierwsze 30 wierszy z pliku *nursery.data*.