Raport laboratorium 4

Cel laboratorium

Celem laboratorium jest zaimplementować algorytm lasu losowego i przeprowadzić klasyfikację dla zadanego zbioru danych.

Założenia

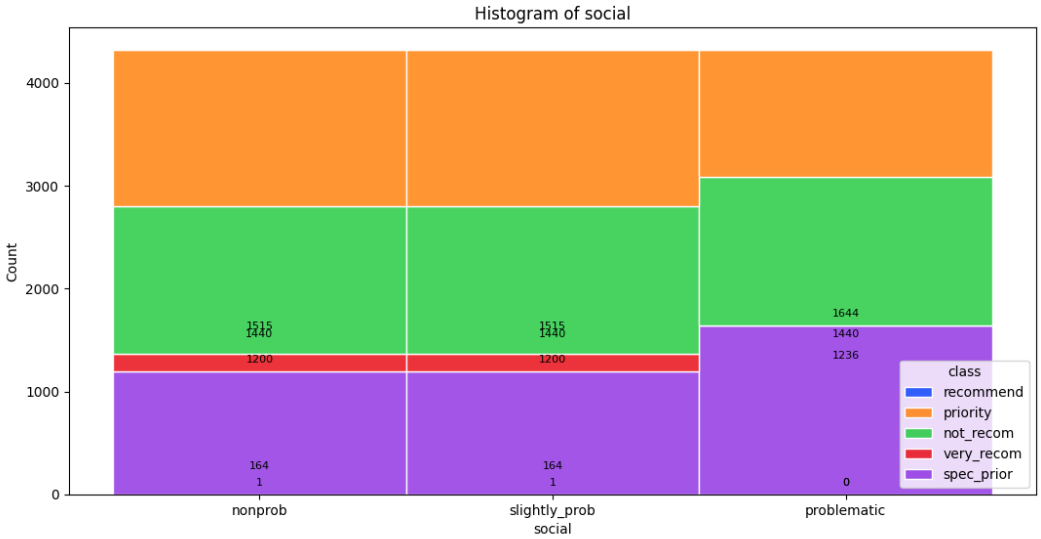
- Zadanie klasyfikacji przyjęcia dzieci do przedszkola na podstawie informacji o strukturze i finansach rodziny. Zbiór tworzy 12960 obserwacji – 5 klas z czego 3 z nich mają podobną liczność (ponad 4000 obserwacji) a 2 są rzadkie (2 i 328 obserwacje).
- W algorytmie lasu losowego został zastosowany algorytm id3 do tworzenia pojedyńczego drzewa
- Wszystkie pomiary zostały wykonane z użyciem tego samego ziarna losowości przy doborze podziału na zbiór trenujący i testujący
- Drzewo losowe w zastosowanym algorytmie id3 generuje tyle poziomów w drzewie ile otrzyma atrybutów w zbiorze D przekazanym jako parametr. Nie występują zatem powtórzenia w węzłach tych samych atrybutów.
- W algorytmie lasu losowego zmieniałem i badałem wpływ głębokości, czyli parametru n_d, który jest ilością atrybutów wylosowanych z głównego zbioru D bez powtórzeń (na zbiór D składają się wszystkie atrybuty występujące w udostępionym pliku nursery.data: `'parents', 'has_nurs', 'form', 'children', 'housing', 'finance', 'social', 'health'`)
- Aby uruchomić program należy w tym samym katalogu umieścić plik nursery.data udostępiony w poleceniu

Biblioteki

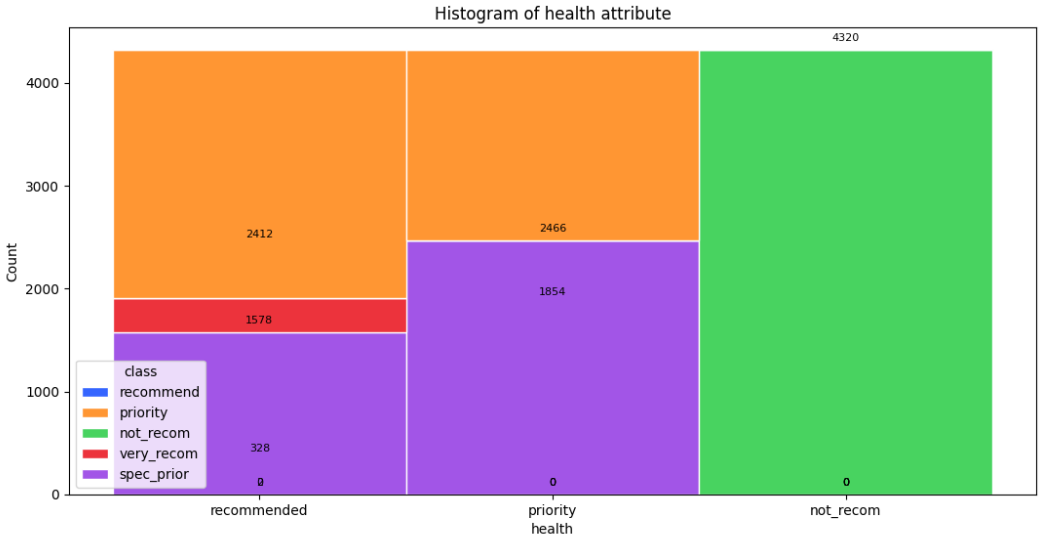
- `math`
- `random`
- `matplotlib`
- `seaborn`
- `sklearn`
- `pandas`

Histogramy - obserwacje

Pierwszy histogram przedstawia rozkład atrybutu social. Widać, że po wartość problematic dużo łatwiej zastosowanemu algorytmowi id3 w drzewie losowym podzielić zbiór, ponieważ występują tylko 3 etykiety klas (priority, non_recom, spec_prior), w przeciwieństwie do wartości atrybutu social: nonprob i slightly_prob, dla których tworzone jest 5 poddrzew na wartości 5 wartości (priority, non_recom, spec_prior, very_recom, recommend).



Drugi histogram przedstawia rozkład atrybutu health. Widać, że wartośc not_recom jest jednoznaczna. Jeżeli ona występuje jest 100% prawdopodobieństwa, że osiągniemy etykiete klasy not_recom. Bardzo dobrze rodziela zbiór również wartość priority, ponieważ dla 4320 wystąpień występują tylko 2 etykiety klas priority i spec_prior. (w podziale 57% wystąpień dla klasy priority i 43% dla klasy spec_prior)



Wpływ sortowania na model

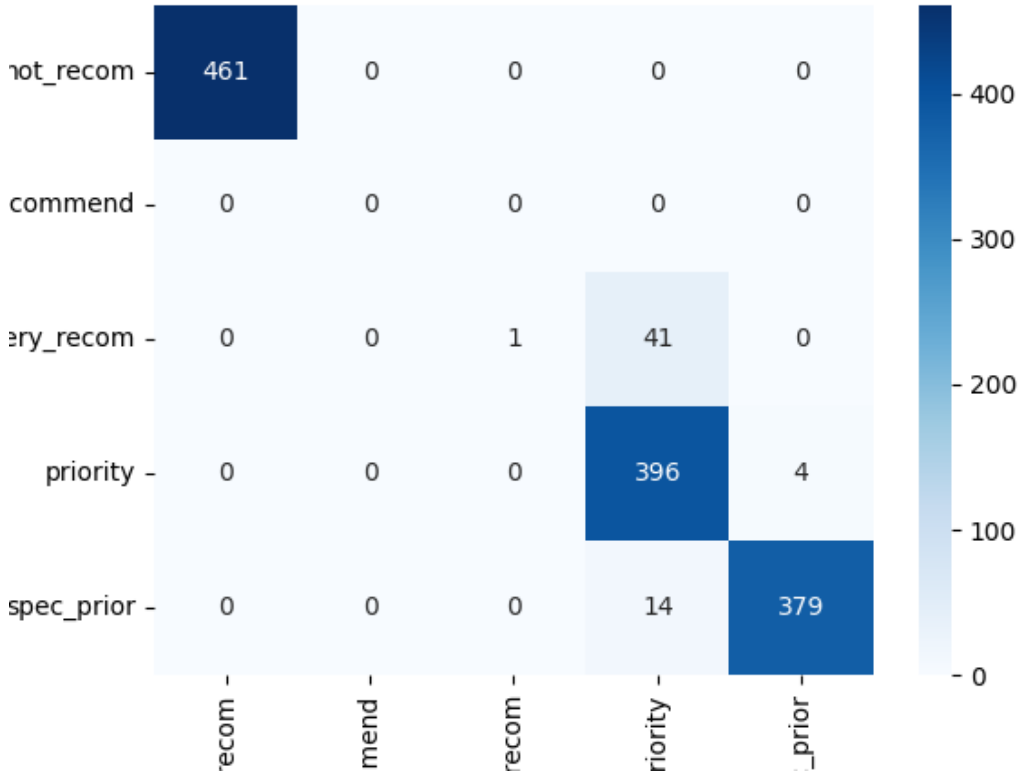
- Jeżeli dane są względnie posortowane, ze względu na dany atrybut bądź klase algorytm działa słabiej, ponieważ przy podziale są wybrane mniej zróżnicowane dane uczące algorytm. Może dojść do takiej sytuacji, w której przypadki niezawarte w próbkach uczących a znajdujące się w próbkach testowych wywołają błędny przydział etykiety przez algorytm.
- W przypadku gdy dane nie są posortowane jest większa szansa, że wszystkie przypadki zostaną uwzględnione przy budowie drzewa decyzyjnego, przez co algorytm będzie działał poprawnie w większej liczbie przypadków.
- Przy takich samych ustawieniach (20% danych testujących, 80% trenujących, n_u = 75%u, n_d = 6) przy posortowaniu danych niezależnie od atrybutu spadek był niewielki na poziomie kilku procent

Pomiary - wpływ głębokości i rozkład wyników

Pomiar 1

Ustawienia:

- 90% to próbki trenujące
- 10% to próbki testujące
- `n_u = int(0.75*len(u))`
- `n_d = 6`

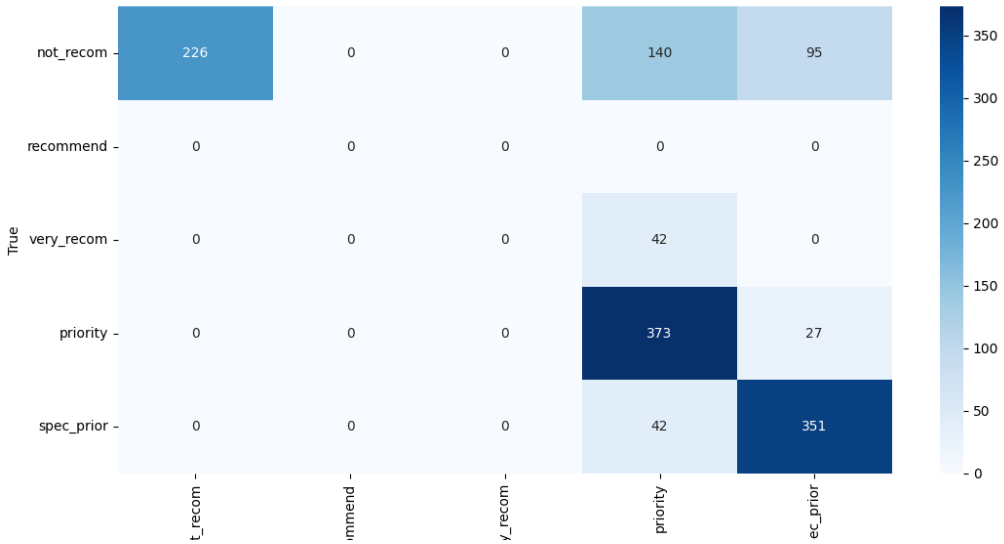


accuracy	precision	recall
0.95	0.96	0.95

Pomiar 2

Ustawienia:

- 90% to próbki trenujące
- 10% to próbki testujące
- `n_u = int(0.75*len(u))`
- `n_d = 3`

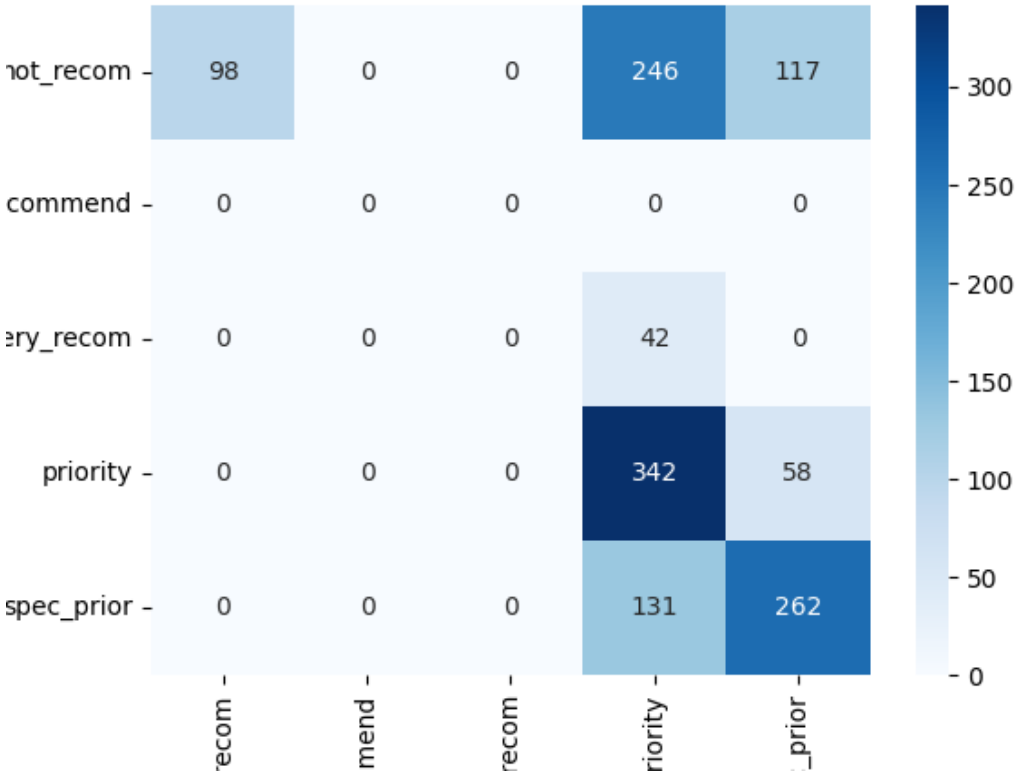


accuracy	precision	recall
0.73	0.77	0.73

Pomiar 3

Ustawienia:

- 90% to próbki trenujące
- 10% to próbki testujące
- `n_u = int(0.75*len(u))`
- `n_d = math.floor(math.sqrt(len(d))) = 2`



accuracy	precision	recall
0.54	0.68	0.54

Wnioski

- Algorytm przy mniejszych głębokościach (parametr n_d) ma gorsze wyniki, ponieważ dochodzi do efektu niedouczenia. Las generuje 100 drzew, jednak jeżeli mają one mało poziomów algorytm nie dokonuje dobrej predykcji etykiet klas. Wraz ze zwrostem głębokości poprawia się działanie algorytmu (od 54% skuteczności aż do 95% przy głębokości 6).
- W zasadzie nie występuje klasa `recommend`, wynika to z dostarczonych danych, ponieważ w zbiorze jest poprostu mało przypadków tej klasy
- Podobna sytuacja występuje w przypadku klasy `very_recom` , również jest to mała klasa, jednak występuje w przeprowadzonych pomiarach. Jest jej na tyle mało, że algorytm nie nauczył się poprawnie przewidywać na podstawie dostarczonego zbioru trenującego.
- Przy większej głebokości budowane drzewa mają więcej poziomów przez co zwiększa się stopniowo czas działania całego algorytmu

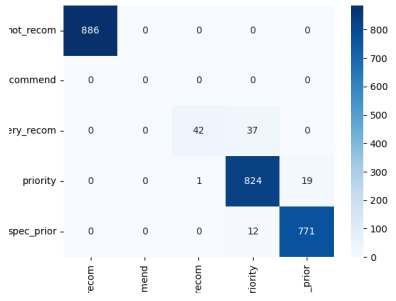
Pomiary - wpływ podziału

Pomiary zostały wykonane dla ustawienia:

- `n_u = int(0.75*len(u))`
- `n_d = 7`

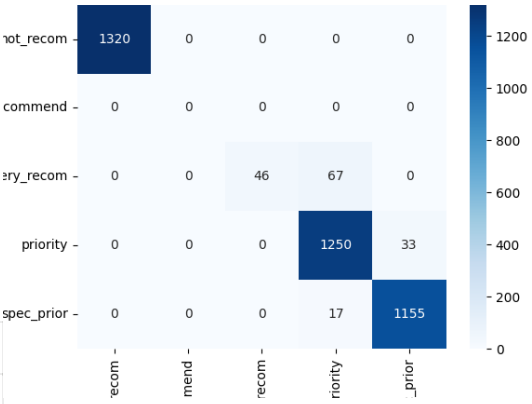
W poszczególnych pomiarach został zmianiany podział danych wejściowych na dane trenujące i testujące:

80% trenujących



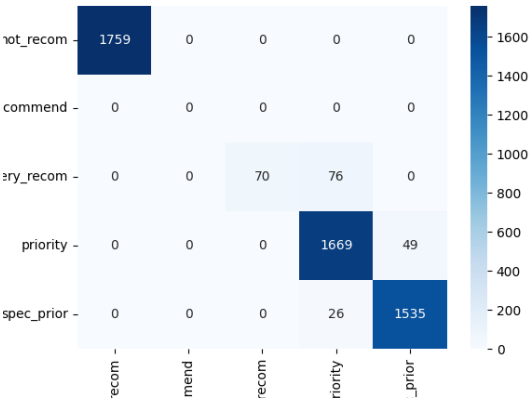
accuracy	precision	recall
0.97	0.97	0.97

70% trenujących



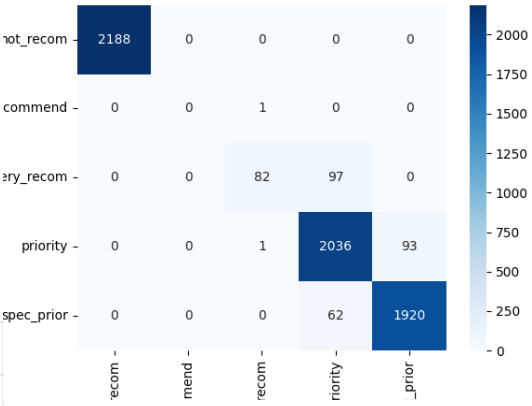
accuracy	precision	recall
0.97	0.97	0.97

60% trenujących

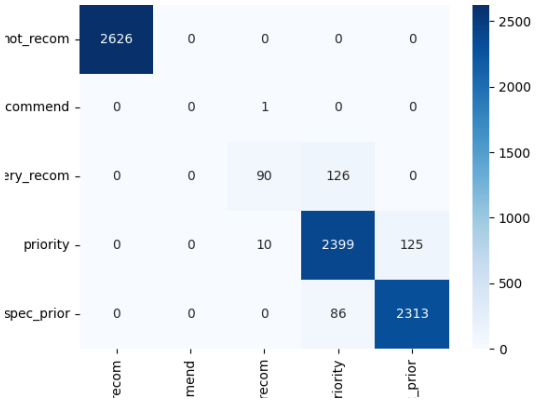


accuracy	precision	recall
0.97	0.97	0.97

50% trenujących

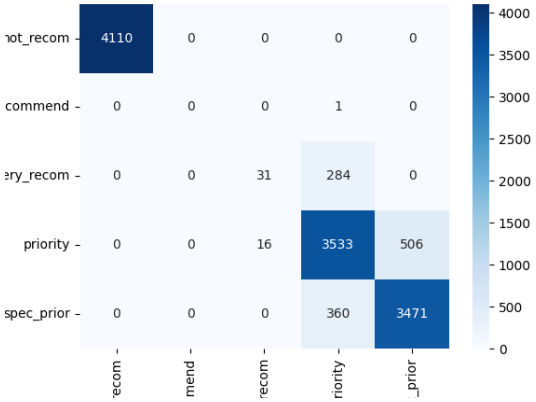


40% trenujących



accuracy	precision	recall
0.95	0.95	0.95

5% trenujących

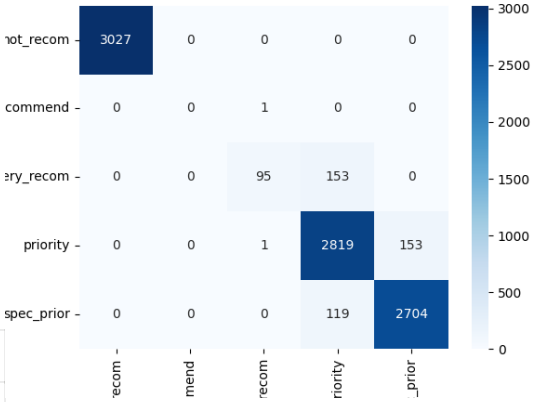


accuracy	precision	recall
0.90	0.90	0.90

0.1% trenujących

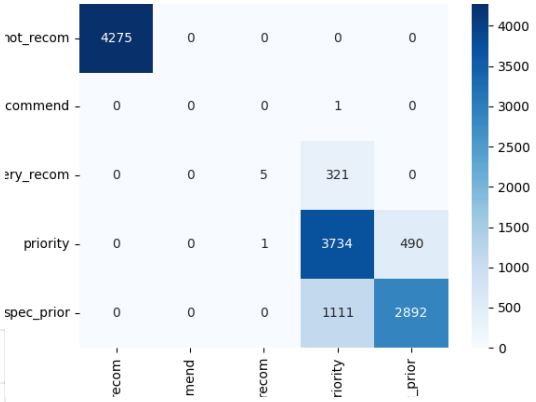
accuracy	precision	recall
0.96	0.96	0.96

30% trenujących

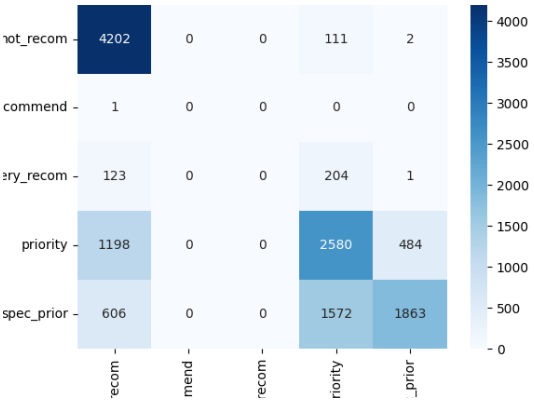


accuracy	precision	recall
0.95	0.95	0.95

1% trenujących



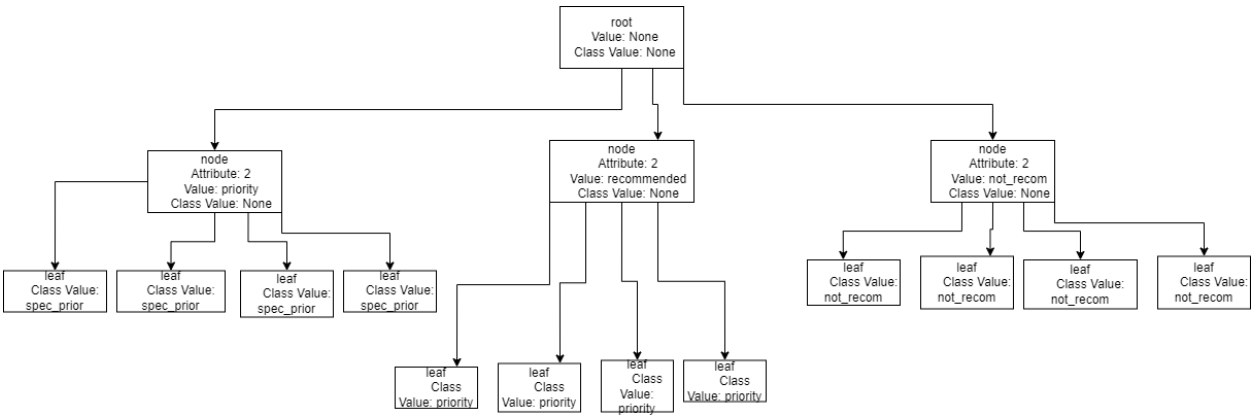
accuracy	precision	recall
0.83	0.83	0.83



accuracy	precision	recall
0.62	0.62	0.62

Wnioski

- Przy stałym zmniejszaniu zbioru trenującego spada skuteczność algorytmu. Dzieje się to jednak bardzo wolno, dopiero na poziomie 5% procent widać znaczący spadek w poprawności działania, na większych zbiorach nie widać drastycznej różnicy. Jest to spowodowane wielkością wyjściowego zbioru, który ma 12960 próbek zatem 5% w tej liczby to wciąż na tyle dużo, że drzewo decyzyjne będzie dobrze wytrenowane osiągając skuteczność dla tego przypadku 90%. Dopiero przy 1% i 0.1% danych trenujących jest odczuwalna przepaść w skuteczności działania. Algorytm wtedy nauczył się na zbyt małej ilości przypadków i przy testowaniu okazuje się, że w zbiorze testującym występują nowe nienaotkane wczęsniej w zbiorze testującym przypadki. Stąd wynika niska skuteczność przewidywania etykiet klas.



Wartości wpisane do diagramu pochodzą z wypisu funkcji print_tree na terminal