

UM Projekt

Eryk Wawrzyn 291018

Semestr lato 2020

Spis treści

1	Temat	2
2	Opis algorytmu podstawowego	2
3	Opis algorytmu uczenia przyrostowego	5
4	Liczenie dopasowania	7
5	Tablice pomyłek	8
6	Test poprawności algorytmu odpowiedzialnego za budowę podstawowego drzewa	9
7	Test poprawności algorytmu odpowiedzialnego za uczenie przyrostowe	10
8	Zbiór „bank”	11
9	Zbiór „agaricus-lepiota”	12
10	Zbiór „iris”	12
11	Współczynnik dopasowanie drzewa oryginalnego w zależności od ilości danych trenujących	13
12	Porównanie algorytmu odpowiedzialnego za drzewo podstawowe i za uczenie przyrostowe	13
13	Wyniki testów dla zbioru „bank”	14
14	Wyniki testów dla zbioru „agaricus-incremental”	15
15	Wyniki testów dla zbioru „iris”	17
16	Złożoność czasowa algorytmu podstawowego	18
17	Złożoność czasowa algorytmu uczenia przyrostowego	19
18	Podsumowanie	19
19	Opis plików	20
20	Uruchomienie projektu	20
21	Zakończenie	20

1 Temat

Drzewo decyzyjne, które potrafi zmieniać się wskutek nadchodzenia nowych danych trenujących (uczenie przyrostowe).

2 Opis algorytmu podstawowego

Na początku należy znaleźć najlepszy test, według którego dane zostaną rozdzielone na dwa zbiory (pierwszy zbiór z danymi spełniającymi warunki testu i drugi zbiór z danymi niespełniającymi warunki testu).

Dla każdego atrybutu należy sprawdzić jaki nastąpiłby podział i jaki zysk informacyjny (niżej wytłumaczenie i sposób liczenia) zostałby uzyskany, jeżeli jako test w danym węźle, użyto by wartość tego atrybutu.

Zysk informacyjny to miara pozwalająca wybrać atrybut, który najbardziej zminimalizuje ilość informacji niezbędnej do klasyfikacji przykładów w partycjach uzyskanych w wyniku podziału.

$$I = GINY(c) - pGINY(f) - (1 - p)GINY(t)$$

I – zysk informacyjny

GINY – współczynnik Giniego

f – zbiór danych niespełniających testu według którego zbiór miałby zostać podzielony

t – zbiór danych spełniających testu według którego zbiór miałby zostać podzielony

c – cały zbiór danych

$$p = \frac{\text{liczba elementów f}}{\text{liczba elementów f} + \text{liczba elementów t}}$$

Współczynnik Giniego określa jak rozproszony jest zbiór danych, im mniejszy tym lepiej, ponieważ wtedy zbiór jest bardziej jednolity (mniej rozproszony).

$$GINY = 1 - \sum \left(\frac{\text{ilość wystąpień danego atrybutu}}{\text{liczba wszystkich elementów}} \right)^2$$

Po przejściu przez wszystkie elementy dziedziny należy dokonać podziału według testu, dzięki któremu uzyskano największy współczynnik informacji i zapisać poniższe elementy:

- test, według którego został dokonany podział;
- uzyskaną wartość współczynnika zysku informacji;
- dane spełniające dany test;
- dane niespełniające dany test;
- liczbę poszczególnych atrybutów;
- następne węzły.

Następnie powyższe operacje należy wykonywać dla każdego węzła, aż zostanie uzyskany zysk informacyjny równy zero.

Prosty przykład

Kolor	Długość ogona	Wartość docelowa
Czarny	Długi	Kot
Biały	Krótki	Kot
Czarny	Krótki	Pies

Korzystając z wzorów z poprzedniej strony obliczany zysk informacyjny dla każdej danej opisującej. Poniżej przykład dla atrybutu „Czarny”.

$$GINY(c) = 1 - \sum \left(\frac{\text{liczba wystąpień danego atrybutu}}{\text{liczba wszystkich elementów}} \right)^2 = 1 - \left(\frac{2}{3} \right)^2 - \left(\frac{1}{3} \right)^2 = \frac{4}{9}$$

$$GINY(f) = 1 - 1 = 0$$

$$GINY(t) = 1 - \left(\frac{1}{2} \right)^2 - \left(\frac{1}{2} \right)^2 = 0,5$$

$$p = \frac{\text{liczba elementów f}}{\text{liczba elementów f} + \text{liczba elementów t}} = \frac{1}{3}$$

$$I = GINY(c) - pGINY(f) - (1 - p)GINY(t) = 0,11$$

Powyższą operację należy wykonać jeszcze dla atrybutów „Biały” (I=0,11), „Długi” (I=0,11), „Krótki” (I=0,11), w tym wypadku wszystkie zyski informacyjne są równe, więc test przyjmuje wartość „Czarny”.

Przykłady zgodne z testem

Kolor	Długość ogona	Wartość docelowa
Czarny	Długi	Kot
Czarny	Krótki	Pies

Przykłady niezgodne z testem

Kolor	Długość ogona	Wartość docelowa
Biały	Krótki	Kot

Procedura jest powtarzana, dla danych niespełniających dany test, I = 0 więc dane są już zostawiane.

Procedura jest powtarzana, dla danych spełniających dany test, atrybut „Długi” (I=0,5), „Krótki” (I=0,5), „Czarny” (I=0), wybrano test „Długi”.

Przykłady niezgodne z testem

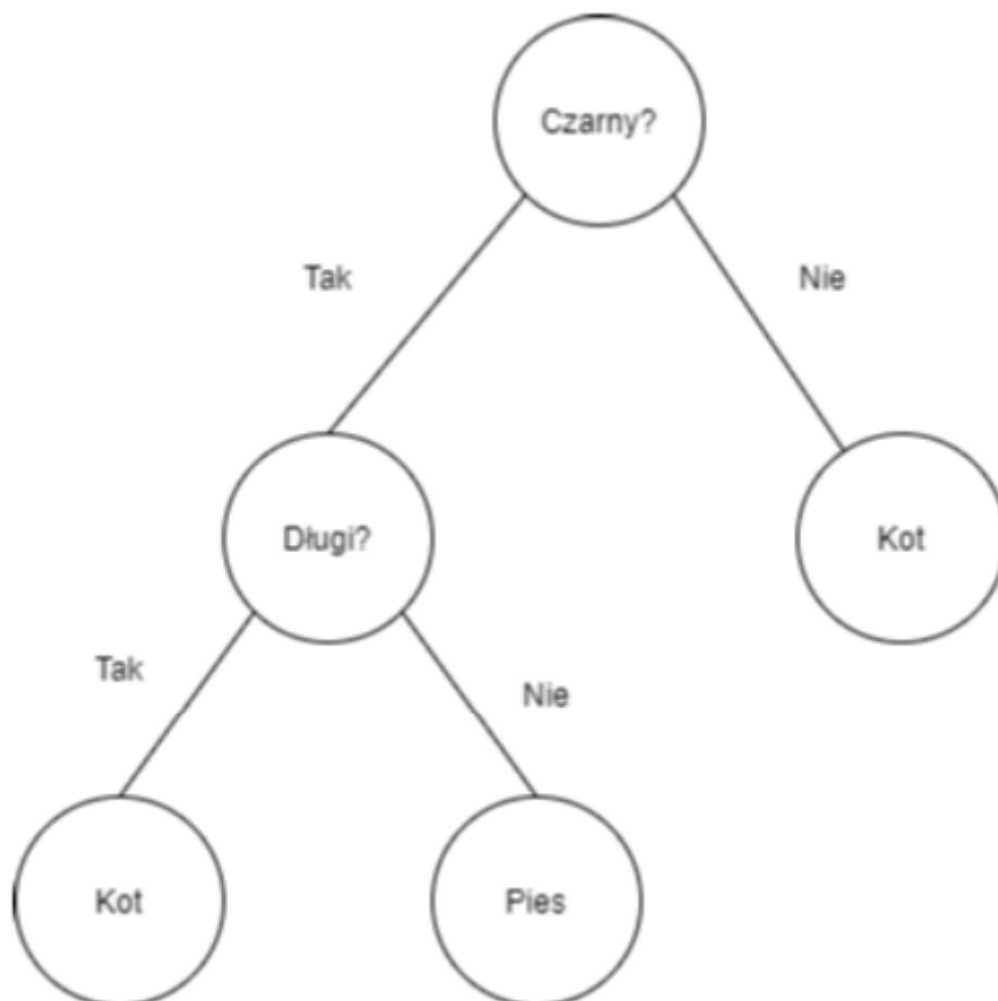
Kolor	Długość ogona	Wartość docelowa
Czarny	Krótki	Pies

Przykłady zgodne z testem

Kolor	Długość ogona	Wartość docelowa
Czarny	Długi	Kot

Procedura jest powtarzana, dla danych spełniających dany test, $I = 0$ więc dane są już zostawiane.

Procedura jest powtarzana, dla danych niespełniających dany test, $I = 0$ więc dane są już zostawiane.



Rysunek 1: Uzyskane drzewo decyzyjne

3 Opis algorytmu uczenia przyrostowego

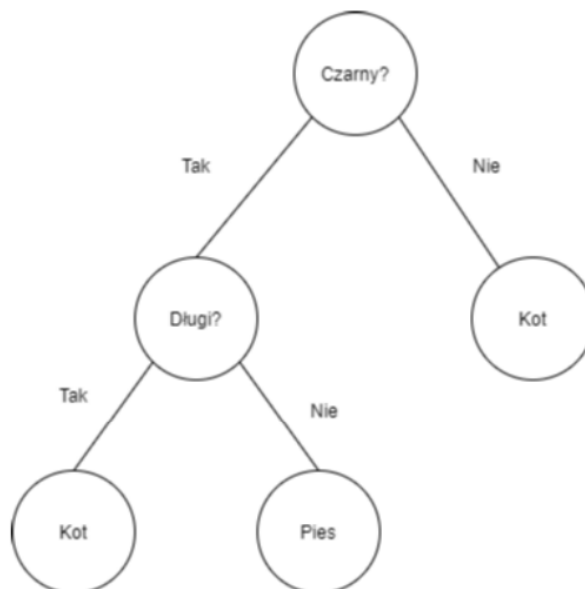
W przypadku gdy zostały dostarczone nowe dane trenujące, zazwyczaj nie opłaca się budować drzewa od nowa podstawowym algorytmem. Wzorowanie na starym drzewie może przyspieszyć douczanie drzewa i pomóc uniknąć gwałtownego spadku dopasowania (wynikającego z nadmiernego dopasowania drzewa do zbioru treningowego).

W celu przyspieszenia algorytmu uczenia przyrostowego zastosowano następujące uproszczenia:

- sprawdzenie czy dane w danym węźle nie występowały w którymś z węzłów stawnego drzewa, jeżeli tak, następuje kopiowanie poddrzewa;
- jeżeli dla testu względem którego został dokonany podział, z węzła starego drzewa jesteśmy w stanie uzyskać dobry zysk informacji, następuje podział według tego testu. Dzięki temu nie trzeba sprawdzać jaki zysk informacji można uzyskać dla każdego możliwego testu, co jest dosyć czasochłonne dla dużej ilości danych;
- jeżeli przez test ze starego drzewa zostanie uzyskany tylko o 0,1 gorszy zysk informacji niż dla najlepszego z możliwych testów, zostaje wybrany stary test (zwiększa to szansę, że w kolejnych węzłach zostaną wykorzystane poddrzewa ze starego drzewa);
- jeżeli, żaden z powyższych warunków nie został spełniony, węzeł jest budowany od nowa (bez wzorowania się na starym drzewie).

Prosty optymistyczny przykład

Kolor	Długość ogona	Wartość docelowa
Czarny	Długi	Kot
Biały	Krótki	Kot
Czarny	Krótki	Pies



Rysunek 2: Drzewo decyzyjne z poprzedniego punktu

Nowe dane

Kolor	Długość ogona	Wartość docelowa
Szary	Długi	Kot
Szary	Krótki	Kot
Szary	Krótki	Kot
Szary	Długi	Kot

Algorytm zaczyna od korzenia, sprawdzane jest czy stare drzewo nie jest zbudowane z takich samych danych, okazuje się, że nie. Zysk informacji po użyciu starego testu, jest bardzo mały. Znajdowany jest test dla którego zysk informacji jest największy, ustawiany jest nowy test "szary", a stare drzewo lub poddrzewo zostaje zapisane.

Przykłady zgodne z testem

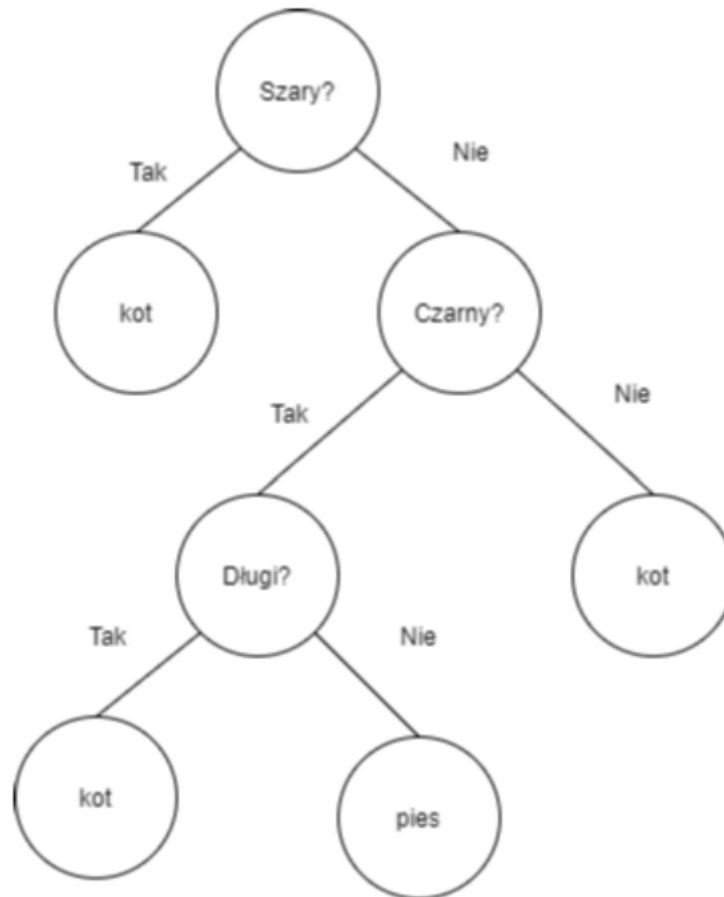
Kolor	Długość ogona	Wartość docelowa
Szary	Długi	Kot
Szary	Krótki	Kot
Szary	Krótki	Kot
Szary	Długi	Kot

Przykłady niezgodne z testem

Kolor	Długość ogona	Wartość docelowa
Czarny	Długi	Kot
Biały	Krótki	Kot
Czarny	Krótki	Pies

Procedura jest powtarzana, dla danych spełniających dany test, $I = 0$ więc dane są już zostawiane.

Procedura jest powtarzana, dla danych niespełniających dany test, sprawdzane jest czy w starym drzewie był taki węzeł z takim testem i z takim zbiorem danych okazuje się, że tak, więc wstawiane jest stare drzewo (lub poddrzewo) w miejsce tego węzła i algorytm kończy działanie.



Rysunek 3: Drzewo decyzyjne po dotrenowaniu

Uzyskano nowe drzewo w trzech krokach, a nie w siedmiu jak by to miało miejsce w przypadku budowy drzewa od początku.

4 Liczenie dopasowania

Atrybuty ze zbioru testowego przechodzą przez węzły zgodnie z testami. Gdy dotrą do liści następuje liczenie dopasowania za pomocą poniższego równania.

$$\text{Dopasowanie} = \frac{Z}{Z + NZ}$$

Z – liczba elementów, dla których hipoteza była spójna z pojęciem docelowym

NZ - liczba elementów, dla których hipoteza nie była spójna z pojęciem docelowym

Im większe dopasowanie, tym algorytm dobiera lepszy zbiór hipotez, ponieważ więcej hipotez zgadza się z pojęciami docelowymi.

5 Tablice pomyłek

KP – klasa predykowana

KR - klasa rzeczywista

KP/KR	Klasa 1	Klasa 2	Klasa 3	Precyzja
Klasa 1	E11	E21	E31	P1
Klasa 2	E12	E22	E32	P2
Klasa 3	E13	E23	E33	P3
Czułość	C1	C2	C3	

Im więcej elementów znajduję się w E11, E22 i E33 tym lepiej (większa czułość i precyzja), ponieważ więcej hipotez zgodziło się z pojęciem docelowym.

Precyzja - stopień zgodności wartości rzeczywistej ze średnią arytmetyczną wyników

P1 – precyzja dla predykcji klasa 1

$$P1 = \frac{E11}{E11 + E21 + E31}$$

SW - suma wszystkich elementów

SP - średnia precyzja

$$SP = \frac{P1 * E11 + P2 * E22 + P3 * E33}{SW}$$

Czułość - wartości opisujące zdolność testu do wykrycia badanej cechy.

C1 – czułość dla pojęcia docelowego klasa 1

$$C1 = \frac{E11}{(E12 + E11 + E13)}$$

SC - średnia czułość

$$SC = \frac{C1 * E11 + C2 * E22 + C3 * E33}{SW}$$

6 Test poprawności algorytmu odpowiedzialnego za budowę podstawowego drzewa

```
Budowa podstawowego drzewa dla danych od 20 do 40
{'no': 14, 'yes': 6}
test -0.39665561009259714 z kolumny 1
True:
{'no': 7, 'yes': 6}
test -0.8671450940660096 z kolumny 0
True:
{'no': 5, 'yes': 6}
test -0.016084581439933576 z kolumny 0
True:
{'no': 5, 'yes': 3}
test 0.34653211205010637 z kolumny 3
True:
{'yes': 2}
False:
{'no': 5, 'yes': 1}
test 1 z kolumny 37
True:
{'yes': 1}
False:
{'no': 5}
False:
{'yes': 3}
False:
{'no': 2}
False:
{'no': 7}
```

Rysunek 4: Wynik budowy podstawowego drzewa

Test wykonano dla danych znajdujących się w pliku bank.csv (od 20 do 40). Uzyskane drzewo jest poprawne. Poprawnie zostały dobrane testy i zgodnie z oczekiwaniami został wykonany podział danych.

7 Test poprawności algorytmu odpowiedzialnego za uczenie przyrostowe

```
Uczenie przyrostowe powyższego drzewa dla danych od 60 do 80
{'no': 33, 'yes': 7}
test -0.6425841750162897 z kolumny 3
True:
{'no': 26, 'yes': 7}
test -1.202325530181361 z kolumny 2
True:
{'no': 21, 'yes': 7}
test -0.8671450940660096 z kolumny 0
True:
{'no': 17, 'yes': 7}
test -0.39665561009259714 z kolumny 1
True:
{'no': 9, 'yes': 6}
test 0.34653211205010637 z kolumny 3
True:
{'yes': 4}
False:
{'no': 9, 'yes': 2}
test -0.2322470110835078 z kolumny 2
True:
{'no': 6, 'yes': 2}
test 1 z kolumny 37
True:
{'yes': 1}
False:
{'no': 6, 'yes': 1}
test 1 z kolumny 7
True:
{'yes': 1}
False:
{'no': 6}
False:
{'no': 3}
False:
{'no': 8, 'yes': 1}
test 1 z kolumny 40
True:
{'yes': 1}
False:
{'no': 8}
False:
{'no': 4}
False:
{'no': 5}
False:
{'no': 7}
```

Rysunek 5: Wynik douczenia drzewa z poprzedniego przykładu

Test wykonano dla danych znajdujących się w pliku bank.csv (od 60 do 80) i drzewa z poprzedniego punktu. Uzyskane drzewo jest poprawne. Poprawnie zostały dobrane testy i zgodnie z oczekiwaniami został wykonany podział danych.

8 Zbiór „bank”

1 - age (numeric)
2 - job : type of job (categorical:
"admin.", "unknown", "unemployed", "management", "housemaid", "entrepreneur", "student",
"blue-collar", "self-employed", "retired", "technician", "services")
3 - marital : marital status (categorical: "married", "divorced", "single"; note: "divorced" means
divorced or widowed)
4 - education (categorical: "unknown", "secondary", "primary", "tertiary")
5 - default: has credit in default? (binary: "yes", "no")
6 - balance: average yearly balance, in euros (numeric)
7 - housing: has housing loan? (binary: "yes", "no")
8 - loan: has personal loan? (binary: "yes", "no")
related with the last contact of the current campaign:
9 - contact: contact communication type (categorical: "unknown", "telephone", "cellular")
10 - day: last contact day of the month (numeric)
11 - month: last contact month of year (categorical: "jan", "feb", "mar", ..., "nov", "dec")
12 - duration: last contact duration, in seconds (numeric)
other attributes:
13 - campaign: number of contacts performed during this campaign and for this client (numeric,
includes last contact)
14 - pdays: number of days that passed by after the client was last contacted from a previous
campaign (numeric, -1 means client was not previously contacted)
15 - previous: number of contacts performed before this campaign and for this client (numeric)
16 - poutcome: outcome of the previous marketing campaign (categorical:
"unknown", "other", "failure", "success")
Output variable (desired target):
17 - y - has the client subscribed a term deposit? (binary: "yes", "no")

Utworzyć drzewo określające czy klient subskrybował lokatę terminową (możliwe wyniki y, n).

9 Zbiór „agaricus-lepiota”

```
1. cap-shape: bell=b,conical=c,convex=x,flat=f,
knobbed=k,sunken=s
2. cap-surface: fibrous=f,grooves=g,scaly=y,smooth=s
3. cap-color: brown=n,buff=b,cinnamon=c,gray=g,green=r,
pink=p,purple=u,red=e,white=w,yellow=y
4. bruises?: bruises=t,no=f
5. odor: almond=a,anise=l,creosote=c,fishy=y,foul=f,
musty=m,none=n,pungent=p,spicy=s
6. gill-attachment: attached=a,descending=d,free=f,notched=n
7. gill-spacing: close=c,crowded=w,distant=d
8. gill-size: broad=b,narrow=n
9. gill-color: black=k,brown=n,buff=b,chocolate=h,gray=g,
green=r,orange=o,pink=p,purple=u,red=e,
white=w,yellow=y
10. stalk-shape: enlarging=e,tapering=t
11. stalk-root: bulbous=b,club=c,cup=u,equal=e,
rhizomorphs=z,rooted=r,missing=?
12. stalk-surface-above-ring: fibrous=f,scaly=y,silky=k,smooth=s
13. stalk-surface-below-ring: fibrous=f,scaly=y,silky=k,smooth=s
14. stalk-color-above-ring: brown=n,buff=b,cinnamon=c,gray=g,orange=o,
pink=p,red=e,white=w,yellow=y
15. stalk-color-below-ring: brown=n,buff=b,cinnamon=c,gray=g,orange=o,
pink=p,red=e,white=w,yellow=y
16. veil-type: partial=p,universal=u
17. veil-color: brown=n,orange=o,white=w,yellow=y
18. ring-number: none=n,one=o,two=t
19. ring-type: cobwebby=c,evanescent=e,flaring=f,large=l,
none=n,pendant=p,sheathing=s,zone=z
20. spore-print-color: black=k,brown=n,buff=b,chocolate=h,green=r,
orange=o,purple=u,white=w,yellow=y
21. population: abundant=a,clustered=c,numerous=n,
scattered=s,several=v,solitary=y
22. habitat: grasses=g,leaves=l,meadows=m,paths=p,
urban=u,waste=w,woods=d
```

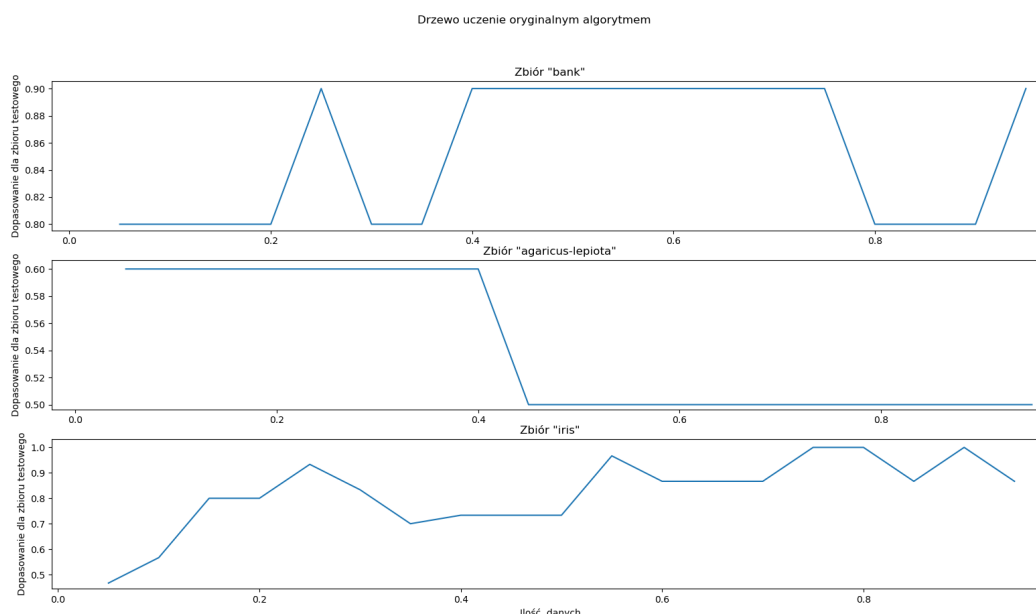
Utworzyć drzewo, które na podstawie danych określa w jakim siedlisku występuje dany grzyb, możliwe wyniki grasses, leaves, meadows, paths, urban, waste, woods (pojęcie docelowe habitat).

10 Zbiór „iris”

```
1. sepal length in cm
2. sepal width in cm
3. petal length in cm
4. petal width in cm
5. class:
  -- Iris Setosa
  -- Iris Versicolour
  -- Iris Virginica
```

Utworzyć drzewo, które na podstawie danych określa gatunek irisa (pojęcie docelowe class).

11 Współczynnik dopasowanie drzewa oryginalnego w zależności od ilości danych trenujących



Rysunek 6: Zależność dopasowania do zbioru testowego od ilości danych (na wykresie na osi x część użytych danych z całego zbioru treningowego, czyli cały zbiór danych bez danych użytych do testów)

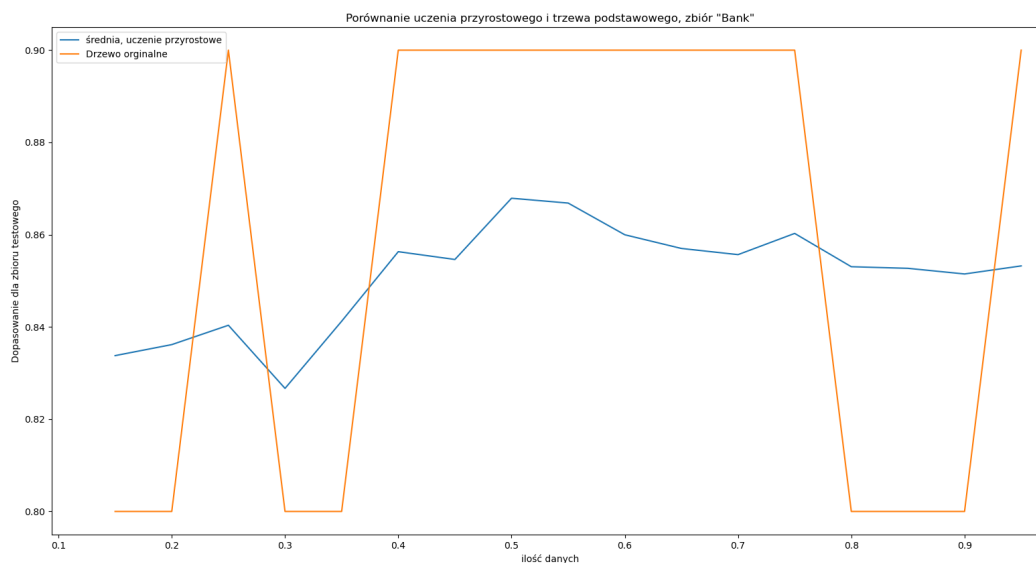
Z przeprowadzonego badania wynika, że ilość danych użytych do trenowania nie ma większego znaczenia. W dopasowaniu drzewa jakie można osiągnąć, bardzo ważne jest z jaką dziedziną pracuje algorytm, jeśli pomiędzy atrybutami występują niewielkie zależności, nie jesteśmy w stanie uzyskać drzewa o dużym dopasowaniu.

Użycie zbyt dużej ilości danych do trenowania, może spowodować nadmierne dopasowanie drzewa do zbioru treningowego. Dlatego warto używać do trenowania różną ilość danych, a na końcu wybrać drzewo najlepiej dopasowane do zbioru testowego. Każdy przypadek należy traktować indywidualnie, dla niektórych zbiorów lepsze będzie użycie do trenowania, większej ilości danych, może się zdarzyć, że warto użyć mniejszego zbioru do trenowania.

12 Porównanie algorytmu odpowiedzialnego za drzewo podstawowe i za uczenie przyrostowe

Średnia uczenia przyrostowego równa jest sumie uzyskanych dopasowań dla ilości danych w poszczególnych trenowaniach równych $n \cdot 0,05 \cdot (\text{ilości danych treningowych})$ i $1 - n \cdot 0,05 \cdot (\text{ilości danych treningowych})$, gdzie $n \in < 1, 19 >$, przez maksymalną wartość n (czyli 19).

13 Wyniki testów dla zbioru „bank”



Rysunek 7: Zależność dopasowania do zbioru testowego od ilości danych dla drzewa oryginalnego i dla średniej wyników uzyskanych dla uczenia przyrostowego (na wykresie na osi x część użytych danych do trenowania z całego zbioru treningowego)

Uzyskane dopasowanie do zbioru testowego dla algorytmu uczenia przyrostowego dla większości badań jest niewiele mniejsze. Jest to spowodowane tym, że algorytm podstawowy szuka testów dla każdego węzła, dla których uzyskany zostanie najlepszy zysk informacji (miara pozwalająca wybrać atrybut, który najbardziej zminimalizuje ilość informacji niezbędnej do klasyfikacji przykładów w partycjach uzyskanych w wyniku podziału). Natomiast w algorytmie uczenia przyrostowego, aby zwiększyć jego szybkość, w pierwszej kolejności zostaje wybierany test ze starego drzewa dla odpowiedniego węzła, jeżeli uzyskany zostanie dla niego dobry zysk informacji.

Z powyższego badania wynika, że w niektórych sytuacjach dzięki algorytmowi uczenia przyrostowego można uzyskać lepsze dopasowanie. Z tego wynika, że nie zawsze warto wybierać test, który daje najlepszy zysk informacji (co ma miejsce w oryginalnym algorytmie). Drzewa budowane za pomocą algorytmu uczenia przyrostowego, dzięki wzorowaniu się na starym drzewie, dają dopasowania o mniejszych odchyleniach standardowych, w zależności od ilości danych użytych w poszczególnych trenowaniach. Algorytm uczenia przyrostowego daje większe szanse na to, że w wyniku douczania współczynnik dopasowania nie spadnie gwałtownie, jak może to mieć miejsce w algorytmie oryginalnym.

Algorytm podstawowy

Ilość danych w zbiorze treningowym = $0.95 \cdot (\text{ilość danych w całym zbiorze} - \text{ilość danych w zbiorze testowym})$.

KP/KR	no	yes	Precyzja
no	743	75	0,91
yes	58	28	0,34
Czułość	0,93	0,30	

Średnia precyzja = 0,86

Średnia czułość = 0,86

Algorytm uczenia przyrostowego

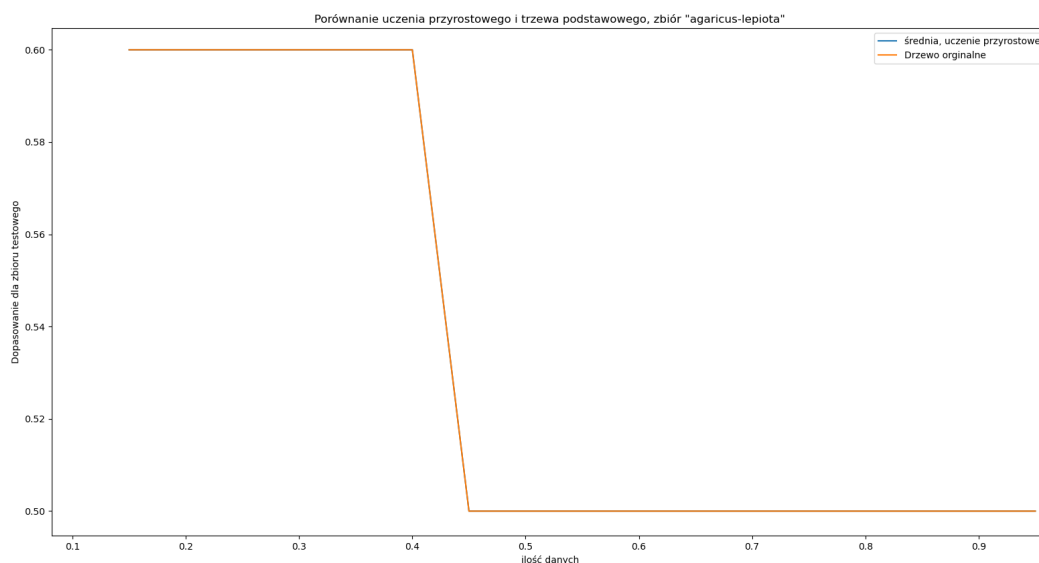
Ilość danych w zbiorze treningowym = $0.95 \cdot (\text{ilość danych w całym zbiorze} - \text{ilość danych w zbiorze testowym})$.

KP/KR	no	yes	Precyzja
no	740	72	0,91
yes	61	31	0,34
Czułość	0,93	0,30	

Średnia precyzja = 0,86

Średnia czułość = 0,86

14 Wyniki testów dla zbioru „agaricus-incremental”



Rysunek 8: Zależność dopasowania do zbioru testowego od ilości danych dla drzewa oryginalnego i dla średniej wyników uzyskanych dla uczenia przyrostowego (na wykresie na osi x część użytych danych z całego zbioru treningowego).

Z przeprowadzonego badania wynika, że dla niektórych zbiorów, obydwa algorytmy dają ten sam wynik. Dla powyższego przykładu dopasowanie zmienia się tylko raz, gdy ilość danych użytych do trenowania przekracza wartość $0.45 \cdot \text{ilość danych w całym zbiorze treningowym}$. Ilość danych użytych do trenowania ma mały wpływ na dopasowanie (poza jednym spadkiem nie ma żadnej różnicy).

Atrybuty opisujące dane pojęcia docelowe mało różnią się od siebie, niezależnie od ilości danych i algorytmu tworzona jest podobna przestrzeń hipotez. Wiele przykładów z dziedziny musi być takich samych lub bardzo podobnych, przez co większa ilość danych treningowych nie wpływa znacząco na uzyskaną przestrzeń hipotez.

Zbiór „agaricus-incremental” posiada bardzo mało atrybutów, powoduje to, że istnieje bardzo mały wybór testów możliwych do zastosowania. Z tego powodu, wyniki analizowanych algorytmów są praktycznie identyczne.

Algorytm podstawowy

Ilość danych w zbiorze treningowym = $0.95 \cdot (\text{ilość danych w całym zbiorze} - \text{ilość danych w zbiorze testowym})$.

KP/KR	u	g	m	p	d	l	w	precyzja
u	21	41	0	0	0	0	0	0,19
g	41	213	53	25	15	0	0	0,61
m	0	51	4	0	0	0	0	0,07
p	0	20	0	4	41	28	0	0,05
d	0	12	0	28	441	17	0	0,91
l	0	0	0	25	28	55	0	0,51
w	0	0	0	0	0	0	49	1
czułość	0,34	0,63	0,07	0,05	0,84	0,55	1	

Średnia precyzja = 0,51

Średnia czułość = 0,50

Algorytm uczenia przyrostowego

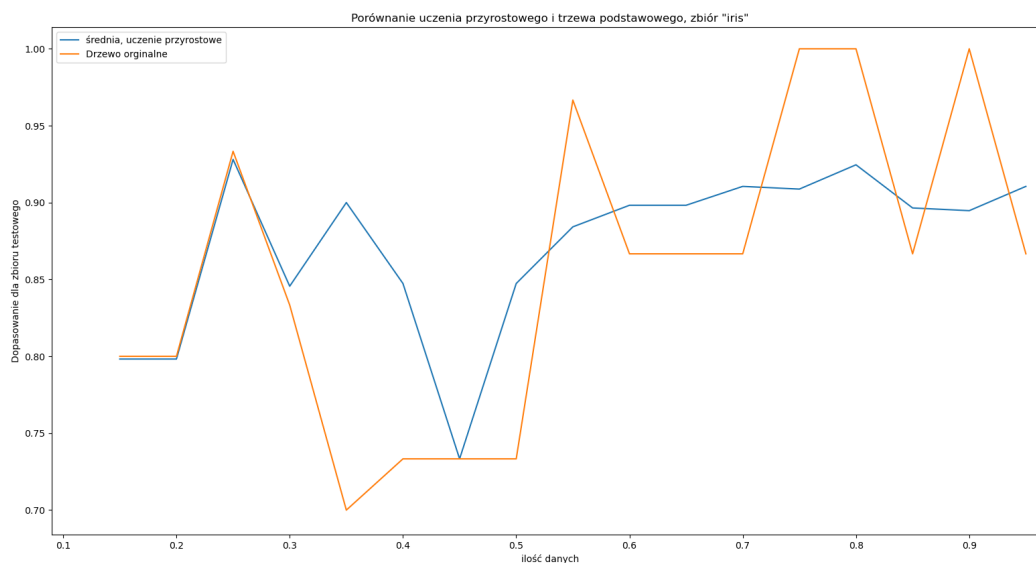
Ilość danych w zbiorze treningowym = $0.95 \cdot (\text{ilość danych w całym zbiorze} - \text{ilość danych w zbiorze testowym})$.

KP/KR	u	g	m	p	d	l	w	precyzja
u	21	41	0	0	0	0	0	0,19
g	40	213	53	26	15	0	0	0,61
m	0	50	3	0	0	0	0	0,07
p	0	20	0	4	41	28	0	0,05
d	0	12	0	27	441	17	0	0,91
l	0	0	0	25	28	55	0	0,51
w	0	0	0	0	0	0	49	1
czułość	0,34	0,63	0,07	0,05	0,84	0,55	1	

Średnia precyzja = 0,51

Średnia czułość = 0,50

15 Wyniki testów dla zbioru „iris”



Rysunek 9: Zależność dopasowania do zbioru testowego od ilości danych dla drzewa oryginalnego i dla średniej wyników uzyskanych dla uczenia przyrostowego (na wykresie na osi x część użytych danych z całego zbioru treningowego)

Z powyższego badania wynika, że dla niektórych dziedzin jesteśmy w stanie uzyskać lepsze dopasowanie dla algorytmu ucznia przyrostowego niż dla algorytmu oryginalnego. Wynika z tego, że wybieranie za każdym razem testu, dla którego uzyskiwany jest najlepszy zysk informacji, nie zawsze prowadzi do uzyskania najlepszej przestrzeni hipotez. Jak widać wzorowanie się na starym drzewie w niektórych sytuacjach jest korzystne.

Algorytm podstawowy

Ilość danych w zbiorze treningowym = $0.95 \cdot (\text{ilość danych w całym zbiorze} - \text{ilość danych w zbiorze testowym})$.

KP/KR	Iris-setosa	Iris-versicolor	Iris-virginica	Precyzja
Iris-setosa	7	0	0	1
Iris-versicolor	0	12	0	1
Iris-virginica	0	4	7	0,64
Czułość	1	0,75	1	

Średnia precyzja = 0,77

Średnia czułość = 0,77

Algorytm uczenia przyrostowego

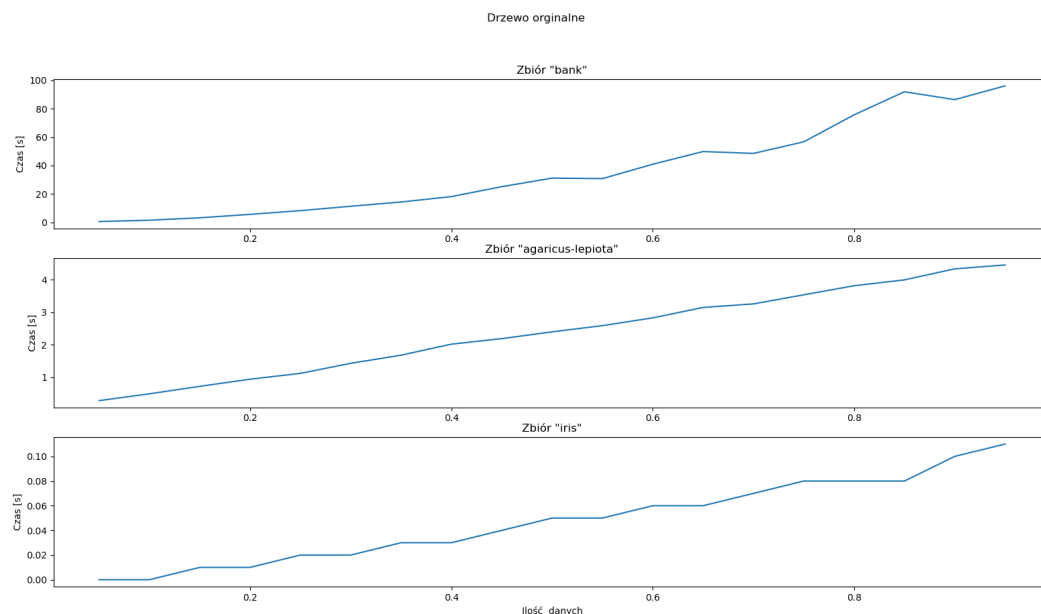
Ilość danych w zbiorze treningowym = $0.95 \cdot (\text{ilość danych w całym zbiorze} - \text{ilość danych w zbiorze testowym})$.

KP/KR	Iris-setosa	Iris-versicolor	Iris-virginica	Precyzja
Iris-setosa	7	0	0	1
Iris-versicolor	0	13	0	1
Iris-virginica	0	3	7	0,7
Czułość	1	0,81	1	

Średnia precyzja = 0,83

Średnia czułość = 0,82

16 Złożoność czasowa algorytmu podstawowego



Rysunek 10: Zależność czasu trwania algorytmu od ilości danych (na wykresie na osi x część użytych danych z całego zbioru treningowego, czyli cały zbiór danych bez danych użytych do testów)

Z powyższego badania wynika, że im większy zbiór treningowy został wykorzystany do trenowania, tym czas trwania algorytmu jest dłuższy. Jeżeli więcej danych przetwarza dany algorytm, wówczas wydłuża się czas wykonywania algorytmu, ponieważ podczas przetwarzania informacji komputer musi wykonać więcej operacji.

17 Złożoność czasowa algorytmu uczenia przyrostowego



Rysunek 11: Zależność czasu trwania algorytmu od ilości danych wykorzystanych w poszczególnych trenowaniach (na wykresie na osi x część użytych danych z całego zbioru treningowego odpowiednio do trenowania podstawowego i do uczenia przyrostowego), wykres pomarańczowy do uczenia przyrostowego użyto oryginalnego algorytmu, wykres niebieski użyto algorytm uczenia przyrostowego.

Z przedstawionego badania wynika, że dla dziedziny, w której znajduje się wiele przykładów takich samych lub bardzo podobnych do siebie lub dany zbiór posiada małą ilość atrybutów (umożliwia to małą ilość testów do wyboru), lepszy jest do douczania algorytm podstawowy. Uzyskiwane jest takie samo dopasowanie, a czas trenowania jest znacznie krótszy. Algorytm podstawowy szuka testu, który daje największy zysk informacji, dzięki temu, gdy dane pozwalają uzyskać mało liczną przestrzeń hipotez, czas jego wykonywania jest krótki. Algorytm uczenia przyrostowego wykonuje sprawdzenia, czy coś można wykorzystać ze starego drzewa, co jest dłuższe niż zbudowanie węzła od nowa, dla dziedziny z atrybutami, mało różniącymi się od siebie.

Dla dziedziny, w której nie znajduje się wiele przykładów takich samych lub bardzo podobnych do siebie, algorytm uczenia przyrostowego okazał się szybszy, w takich sytuacjach korzystniejsze jest wzorowanie się na starym drzewie. Mniej czasochłonne jest sprawdzenie, czy można coś wykorzystać ze starego drzewa niż szukać od nowa testów.

18 Podsumowanie

Używanie algorytmu uczenia przyrostowego w większości przypadków, daje niewiele gorsze dopasowanie, ale w niektórych przypadkach jest znacznie szybsze.

Nie warto używać algorytmu uczenia przyrostowego:

- dla dziedzin w których znajduje się wiele przykładów takich samych lub bardzo podobnych do siebie;
- gdy atrybutów w danej dziedzinie jest bardzo mało.

W przeciwnych wypadkach lepiej używać algorytmu uczenia przyrostowego, czas wykonywania jest zazwyczaj krótszy, a dopasowanie podobne. Algorytm uczenia przyrostowego daje większe szanse, że w wyniku douczania współczynnik dopasowania nie spadnie gwałtownie, jak może to mieć miejsce w algorytmie oryginalnym (przez wzorowanie się na starym drzewie).

Do wyboru algorytmu uczenia przyrostowego lub oryginalnego algorytmu należy podchodzić indywidualnie, w zależności od rodzaju danych i ilości danych. Warto przed trenowaniem całym zbiorem treningowym, zrobić testy na małej dziedzinie, albo sprawdzić czas wykonywania i dopasowanie jakie uzyskiwane jest dzięki algorytmowi.

19 Opis plików

- `Build.py` - plik z algorytmem budowy podstawowego drzewa
- `Incremental_learning.py` - plik z algorytmem uczenia przyrostowego
- `Mesure.py` - plik z funkcjami obliczającymi współczynniki (zysku informacji itp.)
- `Split.py` - plik z funkcjami dokonującymi podziału danych
- `Print_tree.py` - plik z funkcjami do wypisania drzewa
- `Data.py` - plik z funkcją wczytującą dane
- `Data_matching.py` - plik z funkcjami do testów dopasowania
- `Test.py` - plik z funkcją wywołującą testy

Algorytm uczenia przyrostowego i budowy podstawowego drzewa jest uniwersalny, należy jedynie zadbać, aby pojęcia docelowe podczas trenowania znajdowały się w ostatniej kolumnie.

Aby wykonać testy należy uruchomić skrypt o nazwie `Test.py`

20 Uruchomienie projektu

Alby uruchomić projekt należy zainstalować poniższe biblioteki:

- `sklearn-learn`

Instalacja: `pip install -U scikit-learn`

- `pandas`

Instalacja: `pip install pandas`

21 Zakończenie

Dzięki realizacji projektu nauczyłem się:

- podstaw języka python;
- podstaw bibliotek `numpy`, `pandas` i `sklearn-learn`;
- drzew decyzyjnych;
- pracy nad własnymi algorytmami;
- pracy nad zwiększeniem szybkości algorytmu;
- testowanie algorytmów.

Dokumentacja kodu znajduje się w pliku „dokumentacja końcowa - `doxygen.pdf`”