Eryk Wawrzyn 291018

UM Projekt

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

Opis testów i wyników

Aby wykonać niżej opisane testy na swoim komputerze należy uruchomić skrypt o nazwie Test.py.

Liczenie dopasowania

Dane testowe przechodziły przez węzły zgodnie z „zapytaniami”, w chwili gdy dotarły do liścia zostaje liczone dopasowanie za pomocą poniższego równania.

Dopasowanie = Z/(Z+NZ)

Z - Ilość elementów znajdujących się w liściu zgodnych z wartością danej testowej

NZ - Ilość elementów znajdujących się w liściu nie zgodnych z wartością danej testowej

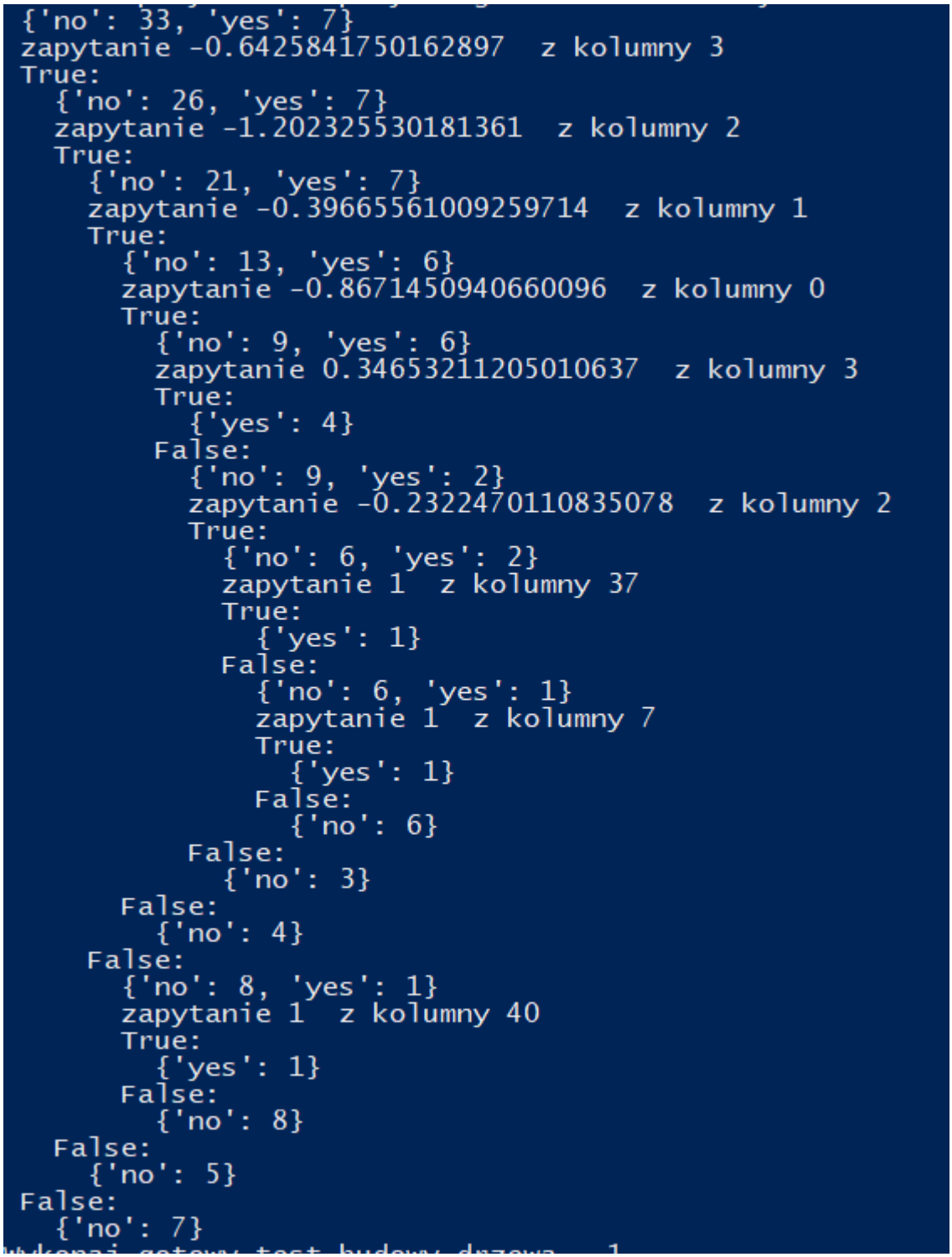
Test poprawności algorytmu odpowiedzialnego za budowę podstawowego drzewa

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

Rys. 1 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 pytania i zgodnie z oczekiwaniami został wykonany podział danych.

Test poprawności algorytmu odpowiedzialnego za przyrostową naukę drzewa



Rys. 2 Wynik douczenia trzewa 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 pytania i zgodnie z oczekiwaniami został wykonany podział danych.

Zadania algorytmu w poszczególnych zbiorach

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.

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

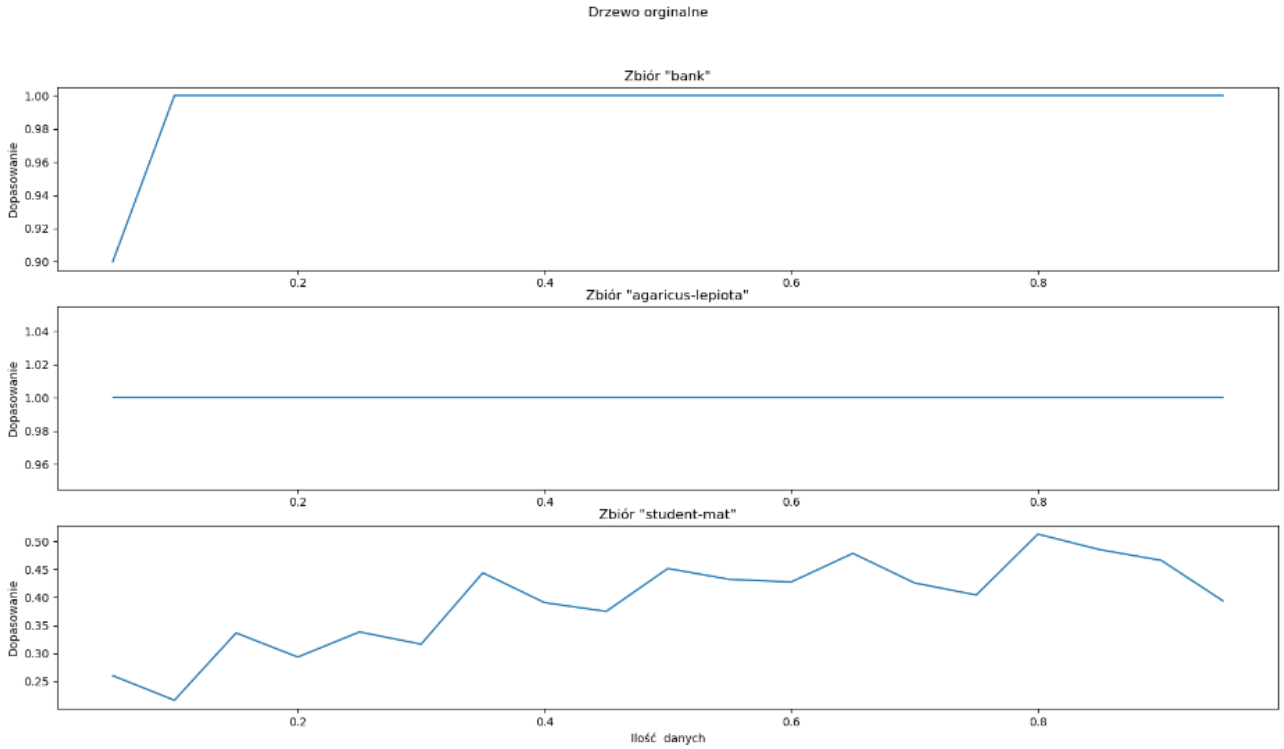
Utworzyć drzewo które na podstawie dachach określa w jakim siedlisku występuje dany grzyb, możliwe wyniki grasses, leaves, meadows, paths, urban, waste, woods (hipoteza habitat)

Zbiór „student-mat”

1 school - student's school (binary: "GP" - Gabriel Pereira or "MS" - Mousinho da Silveira)
2 sex - student's sex (binary: "F" - female or "M" - male)
3 age - student's age (numeric: from 15 to 22)
4 address - student's home address type (binary: "U" - urban or "R" - rural)
5 famsize - family size (binary: "LE3" - less or equal to 3 or "GT3" - greater than 3)
6 Pstatus - parent's cohabitation status (binary: "T" - living together or "A" - apart)
7 Medu - mother's education (numeric: 0 - none, 1 - primary education (4th grade), 2 – 5th to 9th grade, 3 – secondary education or 4 – higher education)
8 Fedu - father's education (numeric: 0 - none, 1 - primary education (4th grade), 2 – 5th to 9th grade, 3 – secondary education or 4 – higher education)
9 Mjob - mother's job (nominal: "teacher", "health" care related, civil "services" (e.g. administrative or police), "at_home" or "other")
10 Fjob - father's job (nominal: "teacher", "health" care related, civil "services" (e.g. administrative or police), "at_home" or "other")
11 reason - reason to choose this school (nominal: close to "home", school "reputation", "course" preference or "other")
12 guardian - student's guardian (nominal: "mother", "father" or "other")
13 traveltime - home to school travel time (numeric: 1 - <15 min., 2 - 15 to 30 min., 3 - 30 min. to 1 hour, or 4 - >1 hour)
14 studytime - weekly study time (numeric: 1 - <2 hours, 2 - 2 to 5 hours, 3 - 5 to 10 hours, or 4 - >10 hours)
15 failures - number of past class failures (numeric: n if 1<=n<3, else 4)
16 schoolsup - extra educational support (binary: yes or no)
17 famsup - family educational support (binary: yes or no)
18 paid - extra paid classes within the course subject (Math or Portuguese) (binary: yes or no)
19 activities - extra-curricular activities (binary: yes or no)
20 nursery - attended nursery school (binary: yes or no)
21 higher - wants to take higher education (binary: yes or no)
22 internet - Internet access at home (binary: yes or no)
23 romantic - with a romantic relationship (binary: yes or no)
24 famrel - quality of family relationships (numeric: from 1 - very bad to 5 - excellent)
25 freetime - free time after school (numeric: from 1 - very low to 5 - very high)
26 goout - going out with friends (numeric: from 1 - very low to 5 - very high)
27 Dalc - workday alcohol consumption (numeric: from 1 - very low to 5 - very high)
28 Walc - weekend alcohol consumption (numeric: from 1 - very low to 5 - very high)
29 health - current health status (numeric: from 1 - very bad to 5 - very good)
30 absences - number of school absences (numeric: from 0 to 93)
these grades are related with the course subject, Math or Portuguese:
31 G1 - first period grade (numeric: from 0 to 20)
31 G2 - second period grade (numeric: from 0 to 20)
32 G3 - final grade (numeric: from 0 to 20, output target)

Utworzyć drzewo które na podstawie dachach określa w jakiej branży matka danego studenta pracuje, możliwe wyniki teacher, health, care related, civil , administrative, police, at_home, other. Drzewa nie zbudowano dla hipotezy przeznaczonej w założeniach dla tego zbioru (G1, G2, G3), wybrano inna hipotezę aby zbadać jak algorytmy zadziałają z danymi mało powiązanymi ze sobą.

Współczynnik dopasowanie drzewa oryginalnego w zależności od ilości danych trenujących

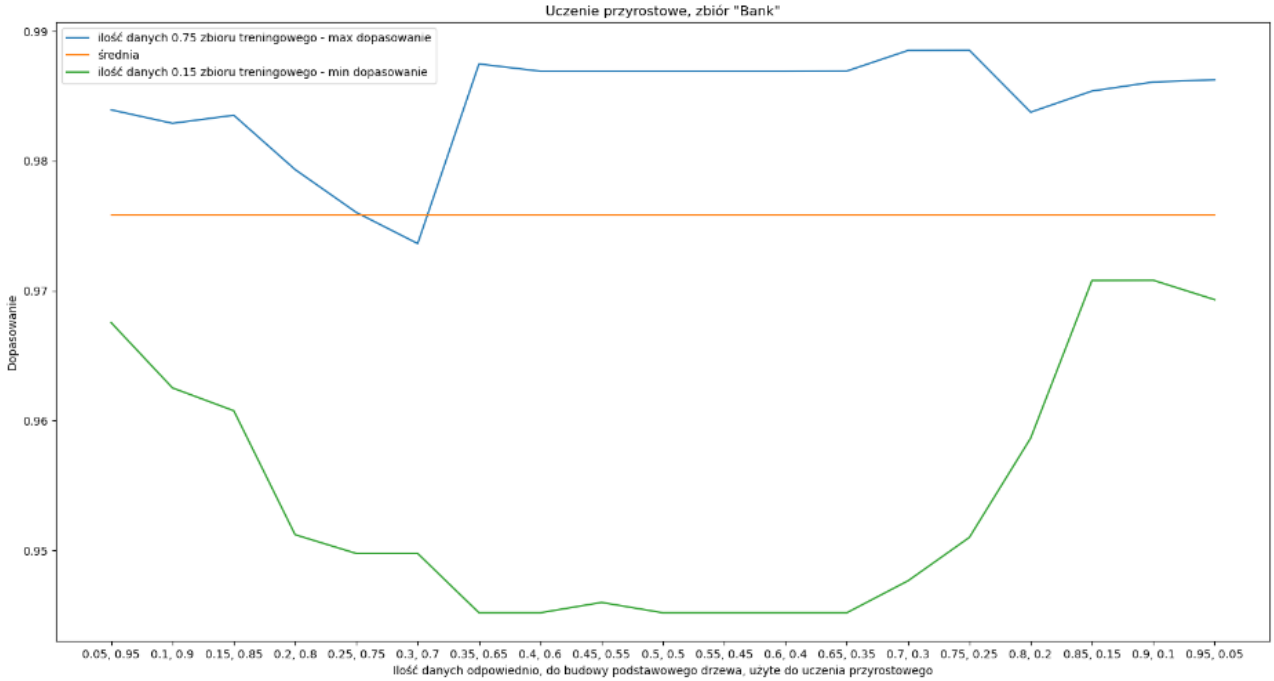


Rys. 3 Zależność dopasowania od ilości danych (na wykresie na osi x część użytych danych z całego zbioru treningowego, czyli cały zbiór danych bez danych użytych do testów)

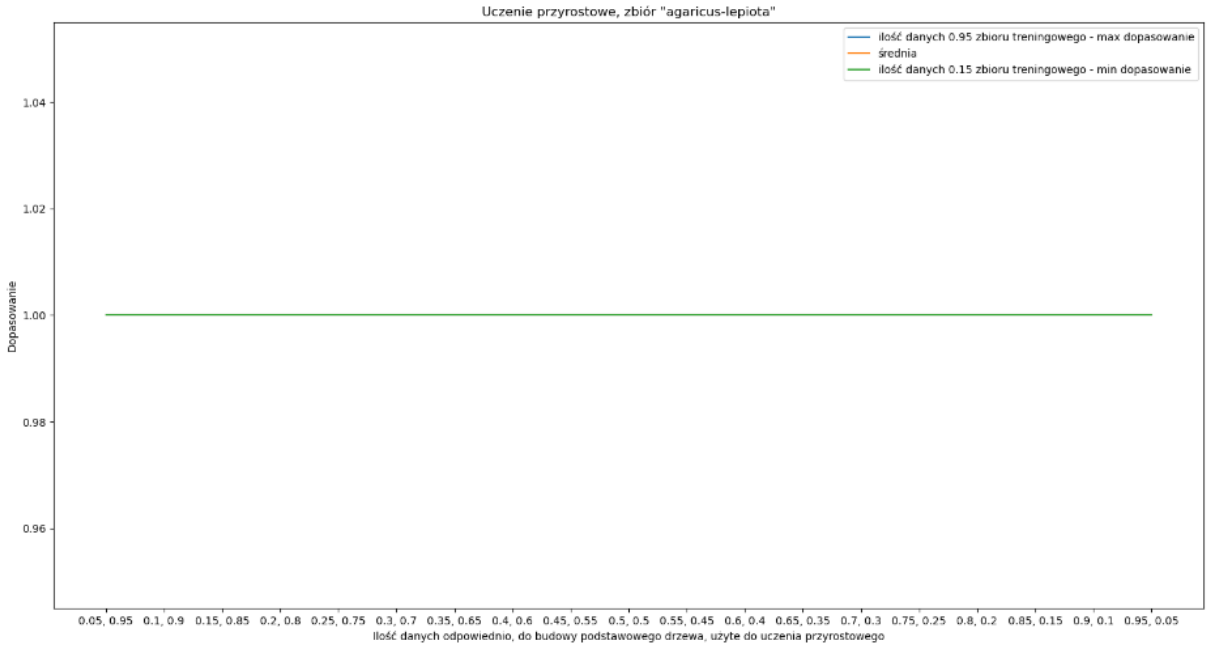
Z przeprowadzonego badania wynika, że ilość danych do trenowania (od pewnej ilości danych) nie ma wielkiego znaczenia. W dopasowaniu drzewa jakie jesteśmy w stanie osiągnąć bardzo ważne jakie dany algorytm przeważa dane, jeżeli nie ma dużego powiązanie pomiędzy danymi, nie jesteśmy w stanie uzyskać drzewa o dobrym dopasowaniu.

Współczynnik dopasowanie drzewa po uczeniu przyrostowym w zależności od ilości danych w poszczególnych trenowaniach

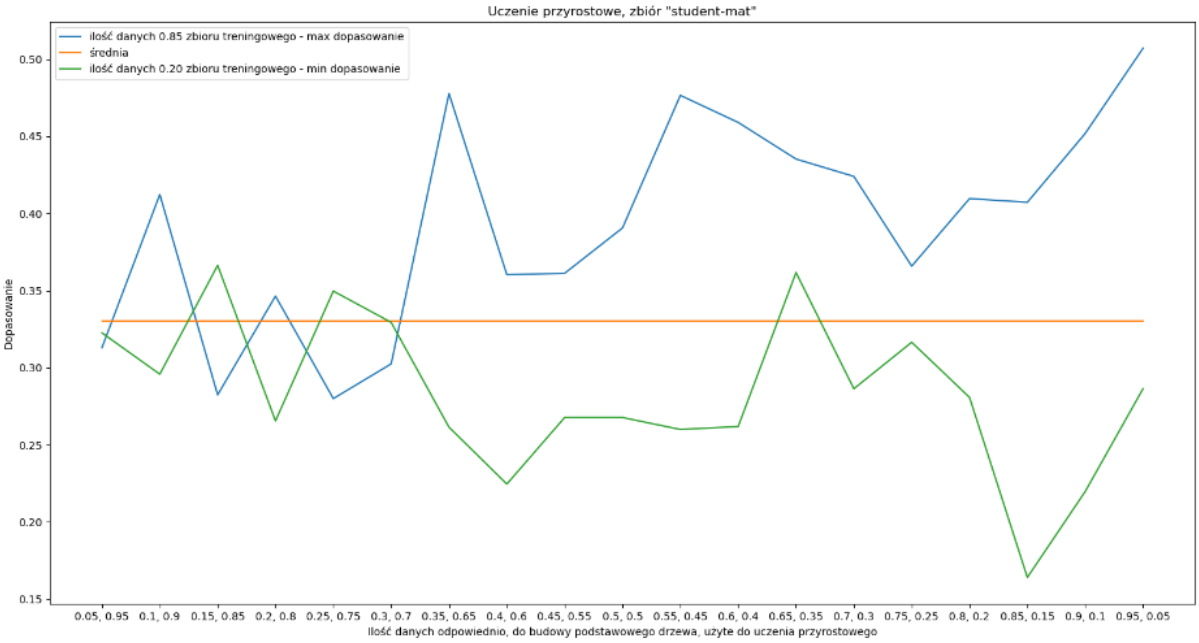
Średnia równa jest sumie uzyskanych dopasowani dla ilości danych trenowaniach równych n*0,05*(ilości danych treningowych), gdzie n$\epsilon < 3, 19 >$, przez maksymalną wartość n odjąć 2 (czyli 17).



Rys. 4 Zależność dopasowania 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)



Rys. 5 Zależność dopasowania 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)

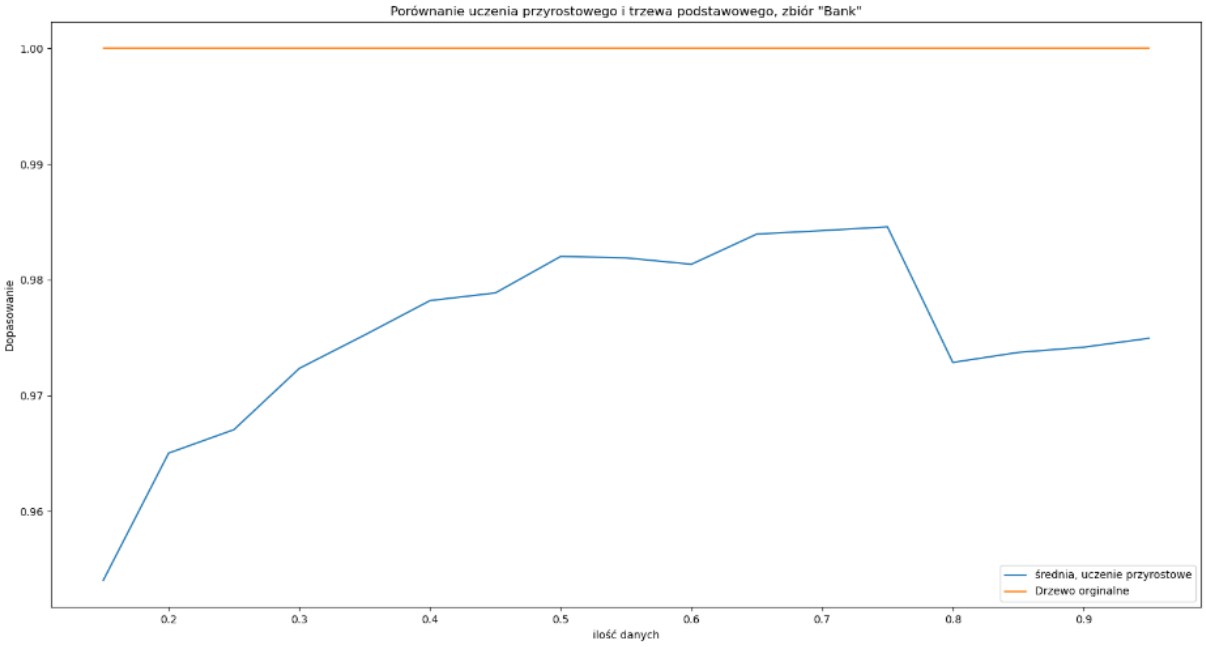


Rys.6 Zależność dopasowania od ilości danych wykorzystanych w poszczególnych trenowaniach (na wykresie na osi x część użytych danych z całego zbioru treningowego odpowiednio do trenowania podstawowego i do uczenia przyrostowego)

Z powyższego badania wynika (szczególnie po średniej), że procent danych wykorzystywanych w poszczególnych trenowaniach ma bardzo małe znaczenie. Tak jak w poprzednim badaniu najważniejsze jest z jakimi danymi pracuje algorytm.

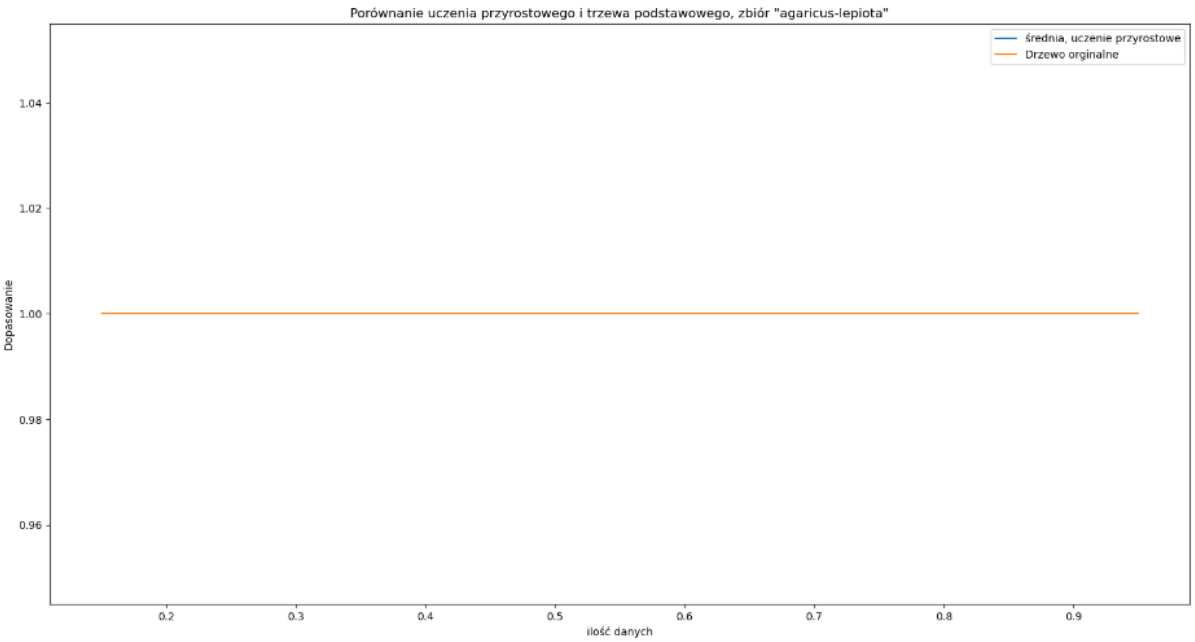
Porównanie algorytmu odpowiedzialnego za drzewo podstawowe i za uczenie przyrostowe

Średnia uczenia przyrostowego równa jest sumie uzyskanych dopasowani dla ilości danych w poszczególnych trenowaniach równych n*0,05*(ilości danych treningowych) i 1 - n*0,05*(ilości danych treningowych), gdzie $n\epsilon < 1, 19 >$, przez maksymalną wartość n (czyli 19).



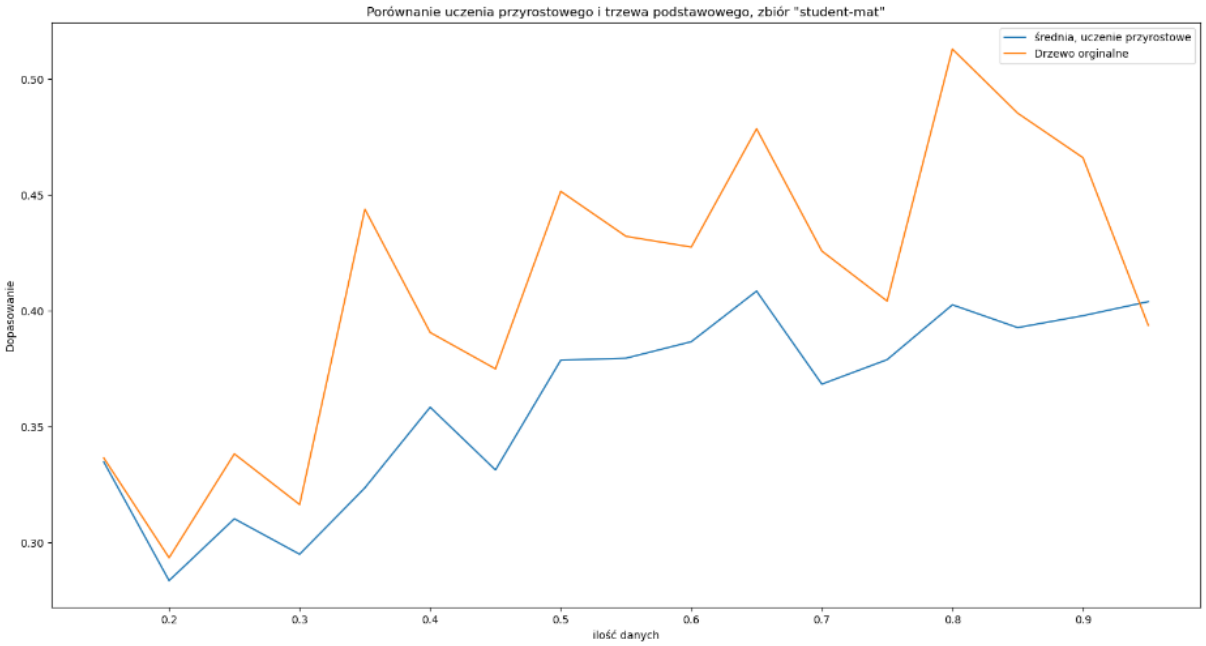
Rys. 7 Zależność dopasowania od ilości danych dla drzewa oryginalnego i dla średniej wyników uzyskanych dla uczenia przyrostowego (na wykresie na osi x część użytych danych do trenowania z całego zbioru treningowego)

Uzyskane dopasowanie dla algorytmu uczenia przyrostowego jest nie wiele mniejsze, jest to spowodowane tym, że algorytm podstawowy szuka jak najlepszej kategorii względem której dane mają zostać podzielone, dla której uzyskamy najlepszy zysk informacji (miara pozwalająca wybrać atrybut, który najbardziej zminimalizuje ilość informacji niezbędnej do klasyfikacji przykładów w partycjach uzyskanych w wyniku podziału), a w algorytmie uczenia przyrostowego aby zwiększyć jego szybkość w pierwszej kolejności wybieramy kategorie ze dla danego węzła starego drzewa, jeżeli uzyskamy dla niego dobry zysk informacji.



Rys. 8 Zależność dopasowania 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)

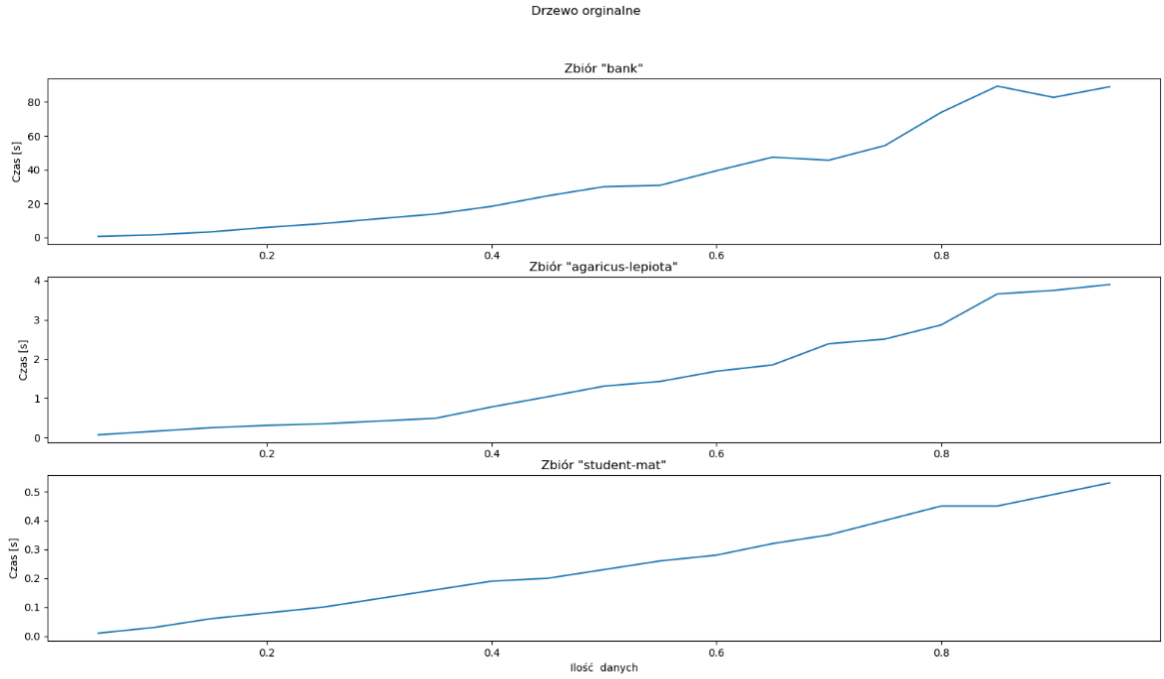
Z powyższego wykresu wynika, że jeżeli algorytm przetwarza dane które mają dużo powiązań między sobą, jesteśmy w stanie uzyskać takie samo dopasowanie w dwóch algorytmach.



Rys. 9 Zależność dopasowania 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)

Jeżeli dane nie pozwalają uzyskać dobrego dopasowania wtedy algorytm uczenia przyrostowego jest znacznie gorszy od oryginalnego algorytmu, ponieważ gdy uzyskujemy słaby zysk informacji w oryginalnym algorytmie i w uczeniu przyrostowym jeszcze „pozwalamy” go zmniejszyć wtedy dopasowanie znacznie się zmniejsza (w wypadku gdy zysk informacji jest bliski jedność i zostanie dozwolone aby zmniejszył się o 0,1 to jest to mała zmiana i uzyskane zostanie dobre dopasowanie, w wypadku gdy jest słaby współczynnik dopasowania i się zmniejszy to dopasowanie znacznie się zmiesza).

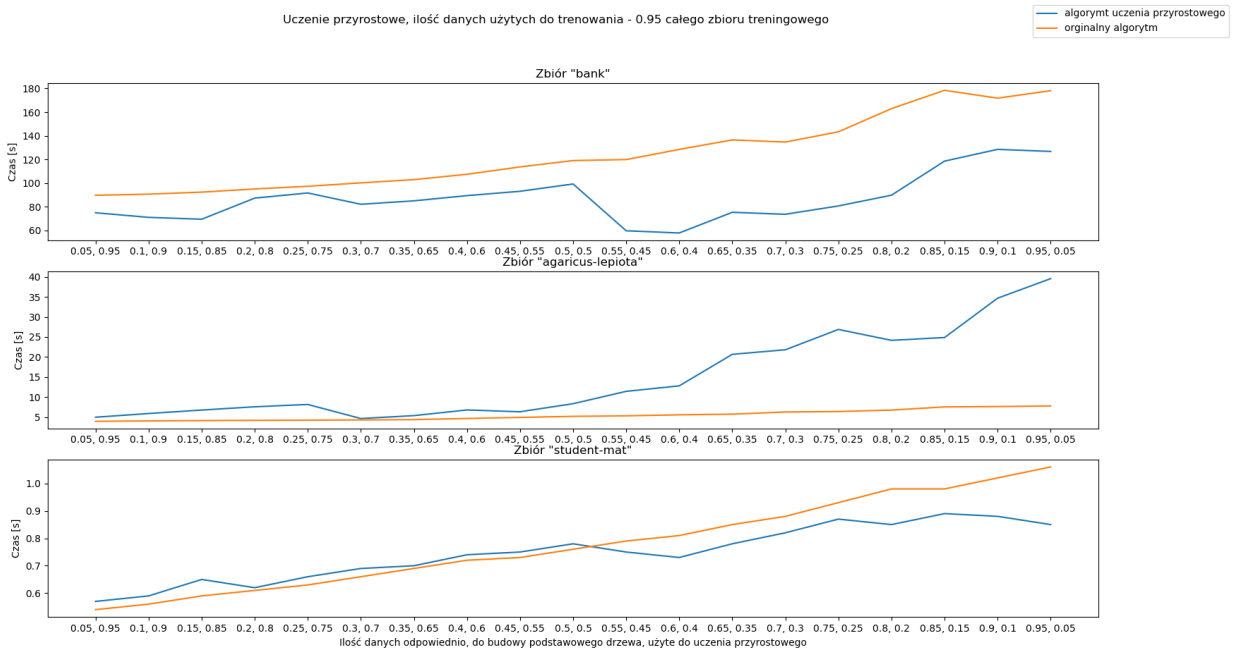
Złożoność czasowa algorytmu podstawowego



Rys. 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 więcej danych zostało wykorzystanych do trenowania tym czas trwania algorytmu jest dłuższy. Im więcej danych przetwarza dany algorytm tym więcej operacji musi wykonać komputer podczas obliczeń, więc czas wykonywania algorytmu się zwiększa.

Złożoność czasowa algorytmu uczenia przyrostowego



Rys. 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 przedstawionych wykresów wynika, że algorytm podstawowy do douczania jest znacznie szybszy od algorytmu uczenia przyrostowego, gdy dane z którymi pracuje algorytm są mocno powiązane ze sobą, algorytm tworzy małe drzewo i musi znaleźć mało kategorii względem którym będą dzielone dane. Natomiast algorytm uczenia przyrostowego jest wolniejszy ponieważ, przy tak dobrze dopasowanych danych, wzorowanie na podstawowym drzewie przedłuża tylko jego budowę, dłużej trwa sprawdzanie czy dany węzeł przyda się przy zbudowanie nowego niż znalezienie nowej kategorii i dokonania podziału bez wzorowania. Dowodzi o tym sytuacja gdy drzewo podstawowe było wytrenowane małą ilością danych (np. 0,05 całego zbioru treningowego), algorytm uczenia przyrostowego musiał dokonać mało sprawdzeń czy coś można wykorzystać z starego drzewa i złożoność czasowa była lepsza. Ulepszeniem obu algorytmów mogło by być sprawdzenie dopasowania przed dotrenowaniem, dla ilości danych równych 0,15 całego zbioru treningowego uzyskano dopasowanie równe jeden, więc żadne dotrenowanie nie było potrzebne.

Dla danych gorzej powiązanych ze sobą, algorytm uczenia przyrostowego okazał się lepszy, w takich sytuacjach lepsze jest wzorowanie się na starym drzewie. Lepsze jest sprawdzenie czy można coś wykorzystać ze starego drzewa, niż szukać od nowa kategorii.

Podsumowanie

Używając algorytm uczenia przyrostowego w większości przypadków daje niewiele gorsze dopasowanie, ale w niektórych przypadkach jest znacznie szybszy.

Kiedy nie warto używać algorytmu uczenia przyrostowego:

- gdy dane nie pozwalają uzyskać dobrego dopasowania dla algorytmu oryginalnego, dla algorytmu uczenia przyrostowego uzyskamy znacznie gorszy współczynnik dopasowania

- w przypadku gdy dane są bardzo mocno powiązane ze sobą (w takim wypadku należy sprawdzić czy w ogóle trzeba douczać drzewo jakimkolwiek algorytmem, może się okazać, że dopasowanie dla starego drzewa dla nowych danych testowych wynosi jeden)

- jeżeli nie ważna jest szybkość algorytmu (zawszę zwiększa to szanse, że trenując drzewo oryginalnym algorytmem zostanie uzyskane większe dopasowanie)

W przeciwnych wypadkach lepiej używać algorytmu uczenia przyrostowego.

Do wyboru algorytmu uczenia przyrostowego lub oryginalnego algorytmu należy podchodzić indywidualnie w zależności od rodzaju danych (jak powiązane są ze sobą), ilości danych.

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 kategorii względem której został dokonany podział, z węzła starego drzewa jesteśmy w stanie uzyskać dobry zysk informacji, następuje podział według tej kategorii. Dzięki temu nie trzewa sprawdzać jaki zysk informacji można uzyskać dla każdej kategorii, które jest dosyć czasochłonne dla dużej ilości danych, ale możemy uzyskać trochę gorszą hipotezę niż było by to możliwe.

- Jeżeli przez starą hipotezę uzyskujemy tylko o 0.1 gorszy zysk informacji niż dla najlepszej kategorii, zostaje wybrane stara kategoria (zwiększa to szanse, że w kolejnych węzłach zostaną wykorzystane poddrzewa ze starego drzewa)

- Jeżeli, żaden z powyższych warunków nie został spełniony to węzeł jest budowany od nowa (bez wzorowania się na starym drzewie).

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

Correctness_of_building.py - plik z funkcjami do testów

Data_matching.py - plik z funkcjami do testów dopasowania

Test.py - plik z funkcją wywołującą testy

W projekcie wykorzystano biblioteki:

- sklearn-learn

Instalacja: pip install -U scikit-learn

- pandas

Instalacja: pip install pandas

Zakończenie

Czego nauczyłem się dzięki realizacji projektu

- podstaw języka python
- podstaw bibliotek numpy, pandas i sklearn-learn
- drzew decyzyjnych
- pracy nad własnymi algorytmami
- pracy nad zwiększeniem szybkości algorytmu
- testowanie algorytmów
- dzięki projektowi zainteresowałem się bardziej machin learningiem i zacząłem uczyć się biblioteki sklearn-learn i planuje w przyszłości tensorflow

Algorytm uczenia przyrostowego i budowy podstawowego drzewa jest uniwersalny, należy jedynie zadbać aby dane względem których będzie dokonywany podział, podczas terowania znajdowały się w ostatniej kolumnie.

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