"Reguły asocjacyjne"

Jan Kwiatkowski

Czerwiec 2023

1 Temat Projektu

Implementacja algorytmu do odkrywanie reguł asocjacyjnych (Apriori, Eclat ,...) pozwalająca na zastosowanie hierarchii elementów m.in. 3 poziomowej. Należy dopuścić reguły a → Ah, gdzie a – jest element w transakcji, Ah - elementem z hierarchii, do którego jest przypisany m.in. element a, jeśli są transakcje t: t ⊃ (a, b), a ∈ E(Ah) b ∈ E(Ah), gdzie E(y) - zbiór elementów przypisanych do elementu y hierarchii.

1.1 Cel

Zbadanie własności zaimplementowanych algorytmów: czas wykonania, uzyskiwane wyniki dla różnych wartości parametrów algorytmu oraz różnej wielkości zbioru wejściowego. Porównanie wyników uzyskanych przez każdy z algorytmów.

2 Podstawowe pojęcia [1]

Reguła asocjacyjna jest wyrażeniem wiążącym dwa rozłączne zbiory pozycji:

$$X \rightarrow Y$$
$$\text{gdzie } \emptyset \neq Y \subseteq I \text{ i } X \subseteq I \setminus Y.$$

O regule X → Y mówi się, że jest oparta na zbiorze pozycji X ∪ Y, przy czym:

- zbiór X ∪ Y jest nazywany bazą reguły X → Y,
- X – jej poprzednikiem,
- Y – jej następnikiem,
- I - zbiór wszystkich elementów.

Wielopoziomową (lub uogólnioną) **regułą asocjacyjną** [2] (ang. multilevel association rule, generalized association rule) są reguły reprezentujące asocjacje pomiędzy grupami elementów (operują na ogólniejszych hierarchiach pojęciowych). WRA nazywamy implikację postaci X → Y, gdzie X i Y zawiera się w zbiorze elementów I oraz zarówno X jak i Y mogą zawierać pozycje z dowolnego poziomu taksonomii T.

3 Przegląd literatury

W ramach projektu zapoznano się także z różnymi artykułami związanymi z generowaniem częstych reguł asocjacyjnych. Najciekawsze pozycje, które w dużym stopniu ułatwiły zrozumienie problemu to:

- 'Mining Generalized Association Rules', R. Srikant, R Agrawal, 1995
- 'Discovery of Multiple-Level Association Rules from Large Databases', J. Han, Y. Fu, 1995
- 'Mining Multiple-Level Association Rules in Large Databases', J. Han, Y. Fu, 1999
- 'Multilevel Association Rules', M. Deshmukh, M. Nashipudimath, January 2014

4 Zbiór danych

Zbiór danych Fruithut pochodzi ze strony:
https://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php

Fruithut

- hierarchiczny, taksonomia 4 poziomy zagnieżdżeń i 43 kategorie
- Jest to zbiór danych zawierający transakcje klientów z amerykańskiego sklepu detalicznego, specjalizującego się w sprzedaży owoców. Zbiór danych zawiera 181 970 transakcji i 1 265 różnych przedmiotów. Największa transakcja zawiera 36 przedmiotów, podczas gdy średnio klient kupuje 3,58 przedmiotów na transakcję.
- transakcje: https://www.philippe-fournier-viger.com/spmf/datasets/fruithut_original.txt
- taksonomia: https://www.philippe-fournier-viger.com/spmf/datasets/Fruithut_taxonomy_data.txt

5 Założenia

Bazując na rozwiązaniach pokazanych w artykułach z rozdziału 3 znaleziono wiele różnych koncepcji implementacji algorytmu rozwiązującego tytułowy problem.

Ostatecznie zdecydowano się na zaimplementowanie algorytmu apriori odpowiednio dostosowanego do odkrywania wielopoziomowych reguł asocjacyjnych.

5.1 Ogólne

W ramach ogólnych założeń opisane zostanie w jaki sposób przygotowano zbiór danych i jakie przyjęto kluczowe wymagania co do tworzenia reguł asocjacyjnych. Można to potraktować jako kroki, które należy wykonać bez względu na wykorzystany algorytm poszukiwania silnych reguł asocjacyjnych.

- Wejściowa baza danych transakcji D oraz taksonomia elementów T (zestaw powiązanych hierarchicznie elementów danych).
- Zbiór transakcji został przekształcony i wczytany do ramki danych. Wiersze oznaczają poszczególne transakcje natomiast kolumny zawierają poszczególne produkty. Jeśli dany produkt wchodzi w skład transakcji to w komórce widnieje wartość 1, jeśli nie to 0.
  Przygotowany zbiór wygląda mniej więcej tak (tabela 1):

Tabela 1: Przygotowany zbiór z transakcjami z przykładowymi danymi (nie rzeczywistymi), wiersze = transakcje, kolumny = produkty.

	1001	1002	1003	1004	1005	1007	1008	1009	1010	...
0	1	0	0	0	1	0	1	1	1	...
1	0	1	0	0	0	1	1	0	1	...
2	0	0	0	1	0	0	0	1	0	...
3	1	0	0	0	0	0	0	1	0	...
4	0	1	0	0	1	0	0	1	0	...
...	...	...	...	...	...	...	...	...	...	...

- Taksonomia elementów po wczytaniu do listy jest przekształcana. Po przekształceniu przechowywana jest w słowniku w następujący sposób:
  - Kluczem jest element z pierwszego poziomu (produkt) a wartością jest lista z kolejnymi, coraz wyższymi poziomami hierarchi (rodzic=>dziadek=>pradziadek).
    Poglądowa zawartość słownika z taksonomią:
    1001: [110, 100], 1002: [150, 100], 1003: [150, 100], 1004: [150, 100], 1005: [130, 100], 1007: [120, 100], 1008: [150, 100], 1009: [120, 100], 1010: [110, 100],...2027: [236, 230, 200], 2028: [236, 230, 200], 2029: [210, 200], 2030: [220, 200], 2031: [236, 230, 200], 2032: [236, 230, 200], 2033: [236, 230, 200],..., 9995: [307], 9996: [307], 9997: [307], 9998: [307]

Nie wszystie elementy mają liczbę poziomów równą 4. Niektóre z produktów mają tylko jednego przodka.

- Do zbioru transakcji dodano informację o hierarchii. Dla dodanych kolumn taksonomii w komórkach przechowywana jest liczności elementów każdej z grup produktów w transakcji. Przedstawia to tabela z rysunku 1.

	1001	1002	1003	1004	1005	1007	1008	1009	1010	1011	...	307	\
0	0	0	0	0	0	0	0	0	0	0	...	0	
1	0	0	0	0	0	0	0	0	0	0	...	0	
2	1	0	0	0	0	0	0	0	0	0	...	0	
3	1	0	0	0	0	0	0	0	0	0	...	0	
4	1	0	1	0	0	0	0	0	0	0	...	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
181966	0	0	0	0	0	0	0	0	0	0	...	0	
181967	0	0	0	0	0	0	0	0	1	0	...	0	
181968	0	1	0	0	0	0	1	0	0	0	...	1	
181969	0	0	0	0	0	0	0	0	0	0	...	0	
181970	0	0	0	0	0	0	0	0	0	0	...	0	

	308	309	310	130	150	210	230	100	200
0	0	0	0	0	1	0	1	1	1
1	0	0	0	0	0	0	1	0	1
2	0	0	0	0	0	0	0	1	0
3	0	0	0	0	0	0	0	1	0
4	0	0	0	0	1	0	0	3	0
...	...	...	...	...	...	...	...	...	...
181966	0	0	0	0	1	0	3	2	4
181967	0	0	0	0	3	1	0	4	1
181968	0	1	0	0	5	1	0	7	1
181969	0	0	0	0	1	0	0	1	0
181970	0	0	0	0	0	0	0	0	0

[181971 rows x 1308 columns]

Rys. 1: Rozszerzony o taksonomię zbiór transakcji.

- Rozszerzone w ten sposób dane są zapisywane do plików 'csv' z odpowiednimi nazwami.
- Częste zbiory są generowane w oparciu o rozszerzony zbiór transakcji taki jak na rysunku 1.
- Wsparcia dla reguły wielopoziomowej, czyli takiej która zawiera co najmniej jeden element z hierarchi są wyznaczane w następujący sposób. Dla podzbioru zawierającego jako jeden z elementów nazwę danej grupy z hierarchi (lub wiecej) to przy wyznaczaniu wsparcia tego podzbioru odnosimy się zawsze do pierwszego poziomu hierarchii (do zbioru transakcji). Jeśli liczba przy rozważanym elemencie (nazwie grupy) wynosi 'Z' to licząc wsparcie dla takiego pozdbioru odnosimy się tylko do transakcji które zawierają wszystkie elementy z pierwszego poziomu w danym podzbiorze oraz tylko nazwy grup o co najmniej 1 większe od wartości 'Z' (czyli jeśli nazwa grupy w analizowanym podzbiorze to owoce i $Z = 2$, to przy liczeniu wsparcia będziemy uwzględniali tylko grupę owoce przy której $Z > 2$).
- Próg wsparcia i zaufania identyczny dla wszystkich reguł.
- Dopuszczalne są wszystkie kombinacje reguł.

- Znajdujemy wszystkie reguły wielopoziomowe dla których:
  - sup > minsup,
  - conf > minconf.

5.2 Algorytm Apriori

Algorytm Apriori został zaimplementowany w klasie AprioriAlgorithm.

1. Konstruktor klasy AprioriAlgorithm przyjmuje minimalne wsparcie (min_support) i minimalną ufność (min_confidence) jako parametry.
2. Dane transakcyjne są wczytywane z pliku za pomocą metody load_data. Możliwe jest wczytanie tylko określonej liczby transakcji poprzez opcjonalny parametr size.
3. Metoda frequency_counting oblicza częstość występowania pojedyńczych elementów oraz częstość występowania elementów taksonomii z danymi wartościami w komórce (liczy częstość wystąpienia każdej wartości z komórki dla danej grupy) w zbiorze transakcji.
4. Metoda generate_frequent_itemsets generuje częste k-elementowe zbiory na podstawie wcześniej obliczonych częstości elementów i kombinacji. Zapamiętuję zbiory rzadkie, żeby przy ich pomocy odrzucić ewentualne kombinacje w kolejnych nadziorach bez konieczności liczenia dla nich wsparcia.
5. Metoda generate_association_rules generuje reguły asocjacyjne na podstawie częstych zbiorów k-elementowych. Wykorzystuje wsparcie, ufność i minimalną liczbę elementów w zbiorze.
6. Algorytm iteruje w pętli, generując kolejne k-elementowe zbiory częste, dopóki istnieją zbiory częste o mniejszej liczbie elementów.
7. W wyniku działania algorytmu, metoda run_algorithm zwraca częstości elementów i kombinacji (transactions_taxonomy_elements_frequency) oraz wygenerowane reguły asocjacyjne (association_rules).

5.3 Algorytm FPGrowth

Algorytm FP Growth został zaimplementowany w pliku fpgrowth_hierarchical.py. Przebieg działania algorytmu fp growth jest nastepujący:

Wejście

- plik csv z rozszerzonymi transakcjami o hierarchie
- plik ze słownikiem taksonomii

Algorytm:

1. Algorytm generuje dane pomocnicze: takie jak słowniki taksonomi na każdym poziomie, słownik dzieci dla każdego rodzica, słownik rodziców dla każdego dziecka.

2. Algorytm tworzy słownik sorted_list, k†óry jest posortowany po wartościach wsparcia itemów malejąco. Jest pomocny w iteracji i tworzeniu drzewa FP.

3. Tworzenie drzewa odbywa się rekurencyjnie w funkcji: constructTree. Iterujemy powszystkich transakcjach i dodajemy do drzewa elewmty zgodnie z kolejnością w sorted_list. Jesli przechodzimy przez element który już odwiedziliśmy zwiekszamy zmienną count w węźle drzewa.

4. Po stworzeniu drzewa przeglądamy pojedyncze itemu tym razem zaczynając od wsparcia najmniejszego. Służy do tego funkcja traverse_from_childs. Przeglądając drzewo tworzone: są Conditional Pattern Base, dodatkowe drzewo Conditional Pattern Tree (używamy funkcji construct_tree). Wynikiem sa zbiory częste freq_items.

5. Ostatnim ktok to generowanie reguł asocjacyjnych. W tym kroku zastosowana jest modyfikacja dla algorytmu hierarchicznego. Podczas liczenia wpsarcia potrzebnego do zaufania liczymy wpsarcia dla itemów z hierarchi wyszukając transakcji z elemntem o jeden więcej w hirarchi. Jest to identyczna modyfikacja jak w algorytmie apriori.

6 Implementacja i uruchomienie

6.1 Zawartość kluczowych plików

W celu poprawnego uruchomienia programu należy odpalić notebooki zgodnie z poniższą kolejnościa.

'preprocess' - pobranie danych 'txt' oraz ich obróbka. Wygenerowanie plików 'csv' z transakcjami, taksonomią i opisami elementów.

'customize_dataset' - wygenerowanie słownika taksonomii oraz rozszerzenie zbioru transakcji o informacje o hierarchii. Wygenerowanie pliku 'transactions_with_parents.csv' na podstawie którego będą potem wyznaczane częste zbiory oraz pliku 'taxonomy_dictionary.csv' zawierającego informacje o słowniku taksonomii.

'apriori' - generowanie częstych zbiorów oraz silnych reguł asocjacyjnych. Na końcu tego pliku znajdują się przeprowadzone eksperymenty.

'pakiet fpgrowth' - zawiera implementacje i eksperymenty dla algorytmu fp growth, znajdje się w pliku /fprgowth/fpgrowth_hierarchical.py

6.2 Format danych wejściowych

Dane wejściowe są pobierane ze strony jako dane w formacie txt odzielone przecinkiem. Działanie programu nie wymaga własnoręcznego pobierania zbioru ani też jakiejkolwiek autoryzacji. Wystarczy wykonać notebook 'preprocess.ipynb'.

6.3 Dodatkowe programy

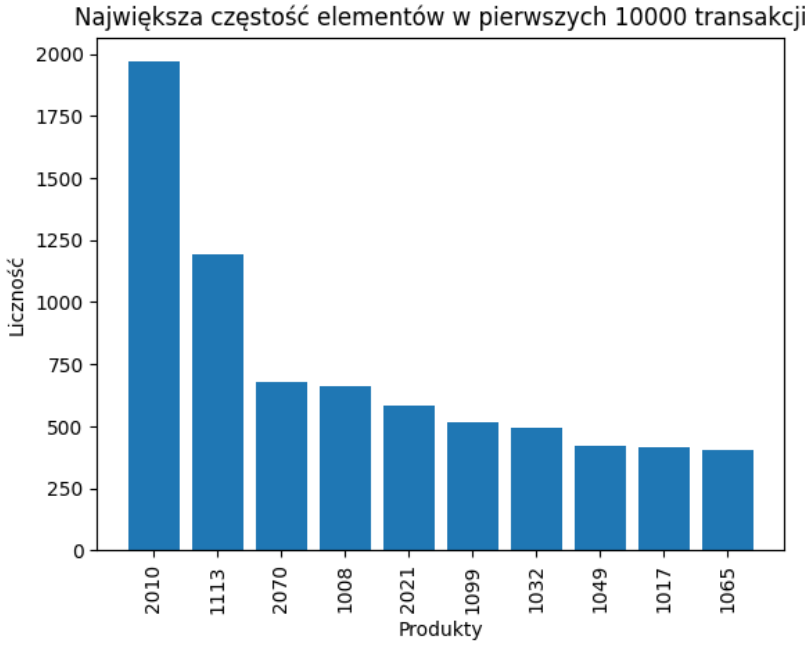
W celu wygenerowania prostego drzewa transakcji oraz taksonomii należy uruchomić notebook o nazwie 'transactions_taxonomy_tree.ipynb'. Do poprawnego uruchomienia wymaga istnienia pliku 'transactions.csv' oraz 'taxonomy_dictionary.csv'.

Nie przesyłano pliku z przeprowadzonymi w trakcie implementacji testami.

7 Eksperymenty, obserwacje i wnioski dla algorytmu apriori.

7.1 Najczęstsze elementy w pierwszych 10000 transakcji

Przeanalizowano częstość występowania produktów dla początkowych 10000 transakcji. Wyniki przedstawione na rysunku 12 mogą sugerować, że to kombinacji tych elementów będziemy spodziewać się w silnych regułach asocjacyjnych a szczególnie produktu o indeksie 2010 który występuje w prawie 20% transakcji.



Rys. 2: Pierwsze 10 najczęstszych elementów w pierwszych 10000 transakcji

7.2 Korelacja produktów

Jeśli dwie zmienne mają wysoką dodatnią korelację, oznacza to, że występuje pozytywna zależność między nimi. W przypadku macierzy transakcji oznacza to, że elementy skorelowane dodatnio mają tendencję do występowania razem w transakcjach. Można oczekiwać, że będą one często występować razem jednak nie musi być to prawda.

Najmocniej skorelowane 10 par produktów dla pierwszych 10k transakcji:
Produkty: ('2080', '4021'), Korelacja: 0.707
Produkty: ('2035', '4001'), Korelacja: 0.577
Produkty: ('1003', '1123'), Korelacja: 0.5
Produkty: ('4070', '7004'), Korelacja: 0.5
Produkty: ('7010', '7012'), Korelacja: 0.471
Produkty: ('2080', '4071'), Korelacja: 0.447
Produkty: ('4001', '4015'), Korelacja: 0.408
Produkty: ('4059', '4060'), Korelacja: 0.408
Produkty: ('6015', '6020'), Korelacja: 0.408
Produkty: ('4052', '4065'), Korelacja: 0.378

7.3 Generowanie reguł

Algorytm apriori generuje silne reguły asocjacyjne na podstawie częstych zbiorów. Jeśli przyjmiemy, że wartość parametru min_sup jest liczbą naturalną stałą dla różnej liczby transakcji wejściowych to wraz ze wzrostem liczby transakcji ilość generowanych reguł będzie coraz większa ponieważ im większy zbiór tym łatwiej znaleźć transakcje, które będą je wspierały.

Inaczej sytuacja wygląda jeśli min_sup jest ułamkiem i wyznaczamy względne minimalne wsparcie zbioru (zadana wartość min_sup na przykład 0.2 mnożona razy liczność zbioru). Oznacza to, że uwzględniamy wtedy liczność transakcji przy określaniu progu min_sup (próg będzie tym większy im większy jest zbiór wejściowy). W takim przypadku liczba częstych zbiorów powinna być podobna dla różnej liczby transakcji oraz ilość wygenerowanych silnych reguł również powinna być podobna.

Dla takiej sytuacji wraz ze wzrostem wartości rMin_Sup liczba wygenerowanych częstych zbiorów i silnych reguł będzie mniejsza.

Wartość parametru min_conf wpływa tylko na ilość wygenerowanych silnych reguł asocjacyjnych (im próg jest większy tym mniej wygenerowanych reguł).

W celu określenia powyższych wniosków przeprowadzono szereg testów jednak w dokumentacji pokazano tylko przykładowy wynik działania algorytmu (stworzonych reguł):

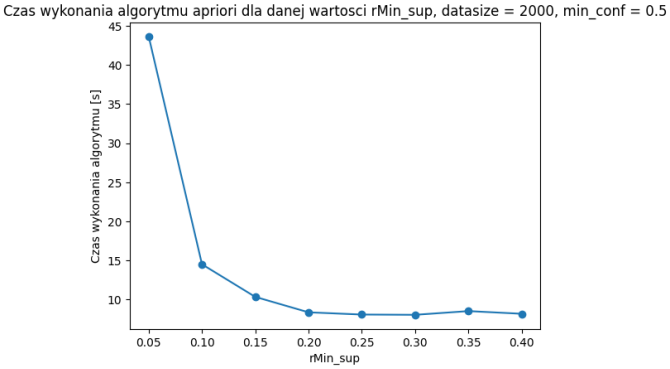
dla dataset_size = 2000, **min_sup = 140**, **min_conf = 0.3** :
X: ('154', 1), ==> Y: 1113, Sup: 174, Conf: 0.534
X: ('150', 1), ==> Y: ('200', 1), Sup: 156, Conf: 0.35
X: ('230', 1), ==> Y: ('200', 2), Sup: 140, Conf: 0.7
X: ('200', 2), ==> Y: ('230', 1), Sup: 140, Conf: 0.598
X: ('220', 1), ==> Y: ('200', 1), Sup: 87, Conf: 1.0
X: ('212', 1), ==> Y: 2010, Sup: 372, Conf: 1.0
X: ('220', 1), ==> Y: 4029, Sup: 161, Conf: 0.346
X: ('210', 1), ==> Y: ('200', 1), Sup: 37, Conf: 1.0
X: ('230', 1), ==> Y: ('200', 1), Sup: 200, Conf: 1.0
X: ('200', 1), ==> Y: ('230', 1), Sup: 200, Conf: 0.366
X: ('132', 1), ==> Y: ('130', 1), Sup: 16, Conf: 1.0
X: ('200', 1), ==> Y: ('100', 1), Sup: 197, Conf: 0.361
X: ('100', 1), ==> Y: ('200', 1), Sup: 197, Conf: 0.31
X: ('150', 1), ==> Y: ('100', 1), Sup: 446, Conf: 1.0
X: ('100', 1), ==> Y: ('150', 1), Sup: 446, Conf: 0.701

X: ('235', 1), ==> Y: ('230', 1), Sup: 22, Conf: 1.0
X: ('210', 1), ==> Y: 2010, Sup: 344, Conf: 0.77

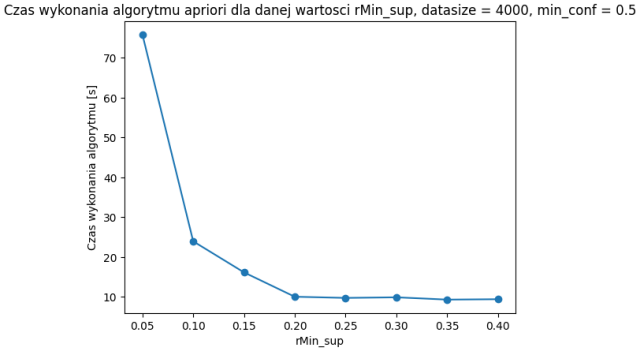
dla dataset_size = 5000, min_sup = 140, min_conf = 0.3 :
X: ('132', 1), ==> Y: ('130', 1), ('100', 2), ('150', 1), Sup: 28, Conf: 0.538
X: ('132', 1), ('150', 1), ==> Y: ('130', 1), ('100', 2), Sup: 28, Conf: 1.0
X: ('130', 1), ('132', 1), ==> Y: ('100', 2), ('150', 1), Sup: 28, Conf: 0.538
X: ('100', 2), ('132', 1), ==> Y: ('130', 1), ('150', 1), Sup: 28, Conf: 0.636
X: ('130', 1), ('132', 1), ('150', 1), ==> Y: ('100', 2), Sup: 28, Conf: 1.0
X: ('100', 2), ('132', 1), ('150', 1), ==> Y: ('130', 1), Sup: 28, Conf: 1.0
X: ('130', 1), ('100', 2), ('132', 1), ==> Y: ('150', 1), Sup: 28, Conf: 0.636
X: ('154', 1), ==> Y: 1113, ('100', 1), ('150', 1), Sup: 17, Conf: 0.773
X: 1113, ('154', 1), ==> Y: ('100', 1), ('150', 1), Sup: 17, Conf: 1.0
X: ('100', 1), ('154', 1), ==> Y: 1113, ('150', 1), Sup: 17, Conf: 0.773
X: ('154', 1), ('150', 1), ==> Y: 1113, ('100', 1), Sup: 17, Conf: 0.773
X: 1113, ('100', 1), ('154', 1), ==> Y: ('150', 1), Sup: 17, Conf: 1.0
X: 1113, ('154', 1), ('150', 1), ==> Y: ('100', 1), Sup: 17, Conf: 1.0
X: ('100', 1), ('154', 1), ('150', 1), ==> Y: 1113, Sup: 17, Conf: 0.773
X: ('230', 1), ==> Y: ('100', 1), ('200', 1), ('150', 1), Sup: 200, Conf: 0.343
X: ('230', 1), ('100', 1), ==> Y: ('200', 1), ('150', 1), Sup: 200, Conf: 0.816
X: ('230', 1), ('200', 1), ==> Y: ('100', 1), ('150', 1), Sup: 200, Conf: 0.343
X: ('230', 1), ('150', 1), ==> Y: ('100', 1), ('200', 1), Sup: 200, Conf: 1.0
X: ('100', 1), ('200', 1), ==> Y: ('230', 1), ('150', 1), Sup: 200, Conf: 0.38
X: ('200', 1), ('150', 1), ==> Y: ('230', 1), ('100', 1), Sup: 200, Conf: 0.487
X: ('230', 1), ('100', 1), ('200', 1), ==> Y: ('150', 1), Sup: 200, Conf: 0.816
X: ('230', 1), ('100', 1), ('150', 1), ==> Y: ('200', 1), Sup: 200, Conf: 1.0
X: ('230', 1), ('200', 1), ('150', 1), ==> Y: ('100', 1), Sup: 200, Conf: 1.0
X: ('100', 1), ('200', 1), ('150', 1), ==> Y: ('230', 1), Sup: 200, Conf: 0.487
X: ('210', 1), ==> Y: ('200', 2), ('230', 1), 2010, Sup: 27, Conf: 0.3
X: ('210', 1), ('230', 1), ==> Y: ('200', 2), 2010, Sup: 27, Conf: 0.9
X: ('200', 2), ('210', 1), ==> Y: ('230', 1), 2010, Sup: 27, Conf: 0.443
X: ('210', 1), 2010, ==> Y: ('200', 2), ('230', 1), Sup: 27, Conf: 0.365
X: ('200', 2), ('210', 1), ('230', 1), ==> Y: 2010, Sup: 27, Conf: 0.9
X: ('210', 1), ('230', 1), 2010, ==> Y: ('200', 2), Sup: 27, Conf: 1.0
X: ('200', 2), ('210', 1), 2010, ==> Y: ('230', 1), Sup: 27, Conf: 0.509

7.4 Porównanie czasu działania algorytmu w zależności od wartości procentowej min_sup

W badanym przypadku (rysunek 3 oraz 4) dla większego datasetu czas liczenia jest ogólnie większy ale charakterystyka przebiegów jest podobna. Jest to związane z tym że już dla procentowej wartości min_sup = 0.2 nie są generowane żadne silne reguły asocjacyjne. Wynika to z faktu, że ze względu na różną wartość progu min_sup uzależnioną od wielkości zbioru wejściowego liczba wygenerowanych częstych zbiorów jest podobna dla różnej liczby wejściowych transakcji.



Rys. 3: Dla zbioru transakcji 2000 oraz zmiennego poziomu min_sup



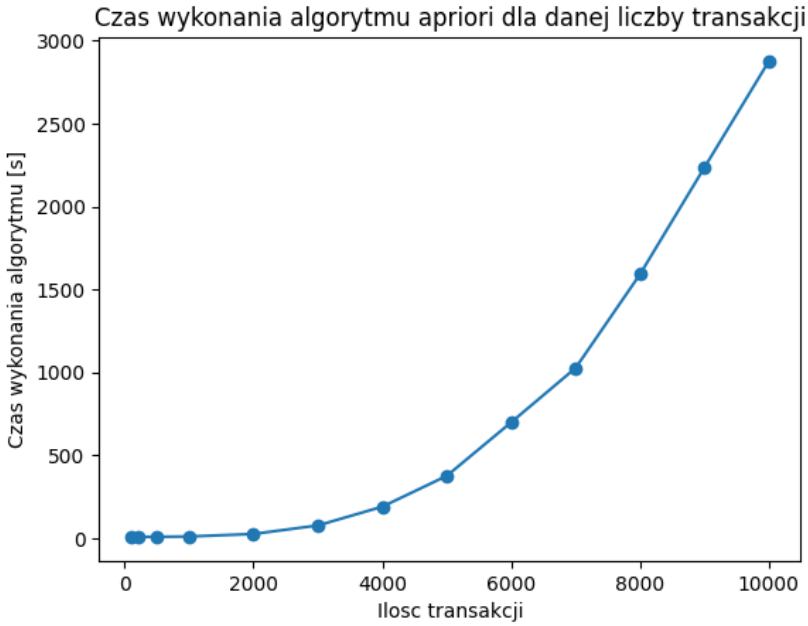
Rys. 4: Dla zbioru transakcji 4000 oraz zmiennego poziomu min_sup

7.5 Porównanie czasu działania algorytmu dla różnej wielkości zbioru wejściowego i jednakowej wartości parametru min_sup = 140 oraz 240

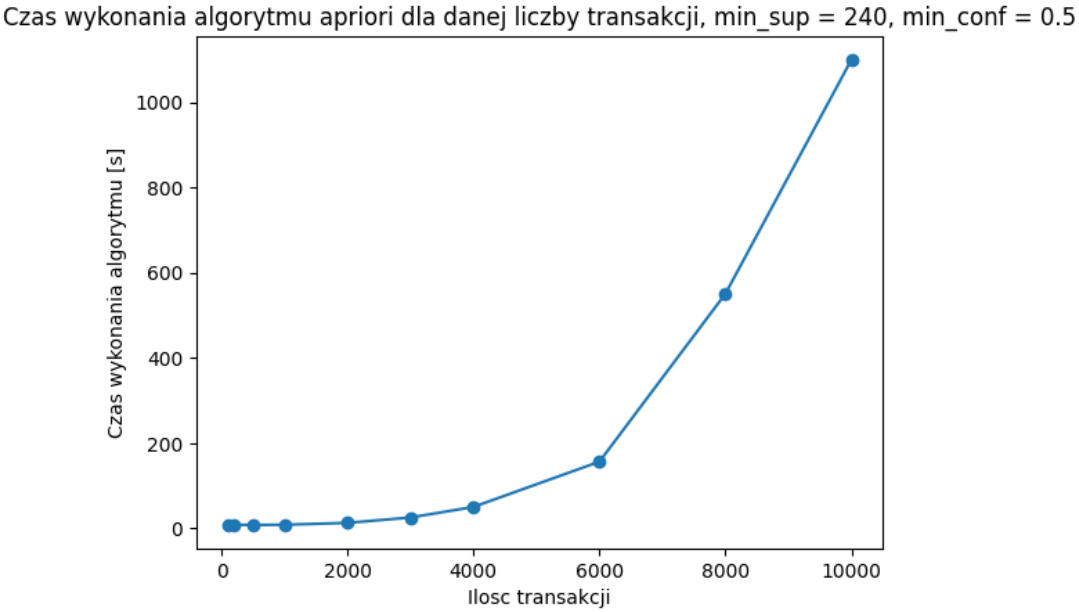
W tym miejscu porównano czas generowania częstych zbiorów oraz wyznaczania silnych reguł asocjacyjnych przez algorytm apriori w zależności od wielkości zbioru wejściowego. Z rysunków 5 oraz 6 widać, że wraz ze wzrostem liczby wejściowych transakcji czas trwania algorytmu rośnie wykładniczo. Już dla 10tys próbek jest to około 50min liczenia, co oznacza że jest to dość wolny algorytm.

W ramach usprawnienia można byłoby użyć drzewa mieszającego (lub drzewa trie) do przechowywania wszystkich zbiorów częstych, jednak dość znaczącym utrudnieniem przy implementacji takiego rozwiązania jest fakt, że w momencie liczenia wsparć dla elementów z poziomu większego niż 1 zawsze musimy odwoływać się do podstawowych transakcji i iterując po całym zbiorze jeszcze raz wyznaczać częstość danych kombinacji uwzględniając przyjęte wcześniej założenia (czyli to że bierzemy tylko transakcje w których liczba elementów z danej grupy jest większa niż cyfra przy nazwie grupy w rozważanej regule).

Wraz ze wzrostem wartości minimalnego wsparcia i minimalnego zaufania zmniejsza się czas pracy algorytmu. Jest to zrozumiałe ponieważ im większa wartość minimalnego progu wsparcia tym mniej częstych zbiorów do przetworzenia, podobnie z zaufaniem reguły.



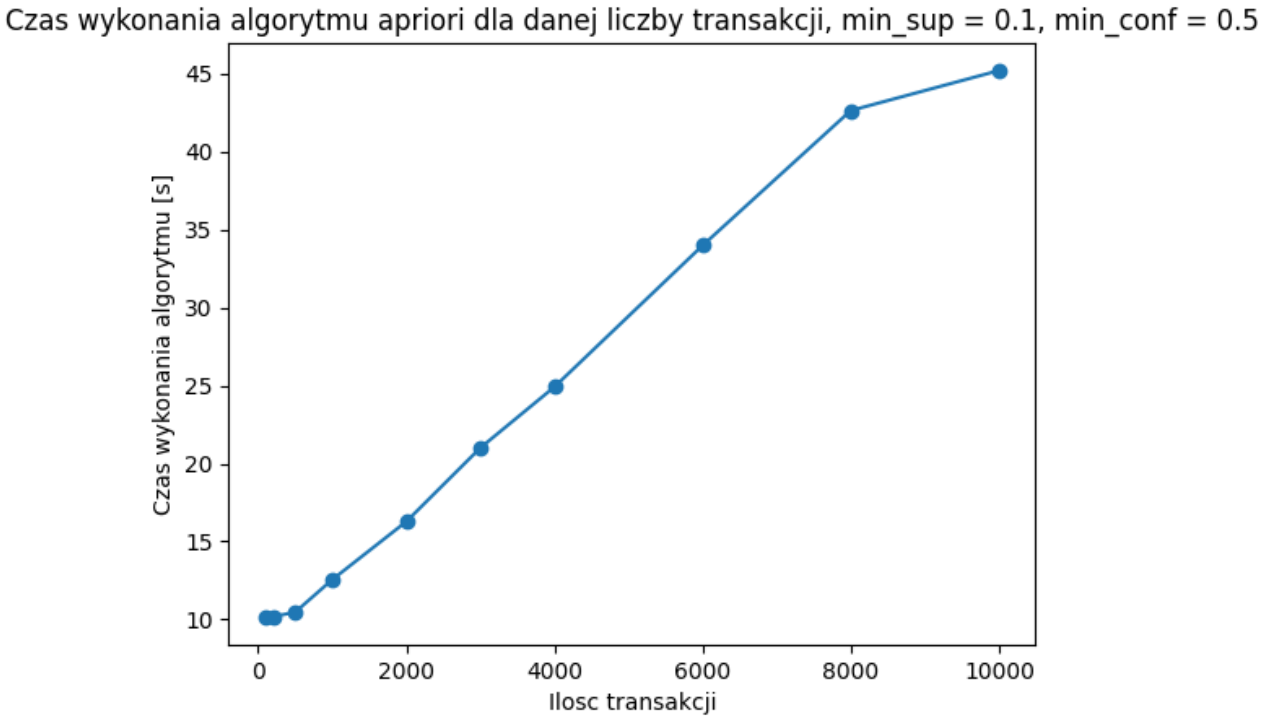
Rys. 5: Czas działania algorytmu od wielkości zbioru wejściowego dla min_sup = 140 i min_conf = 0.3.



Rys. 6: Czas działania algorytmu od wielkości zbioru wejściowego dla min_sup = 240 i min_conf = 0.5.

7.6 Czas działania algorytmu dla różnej wielkości zbioru wejściowego i zmiennej wartości parametru min_sup = 0.1

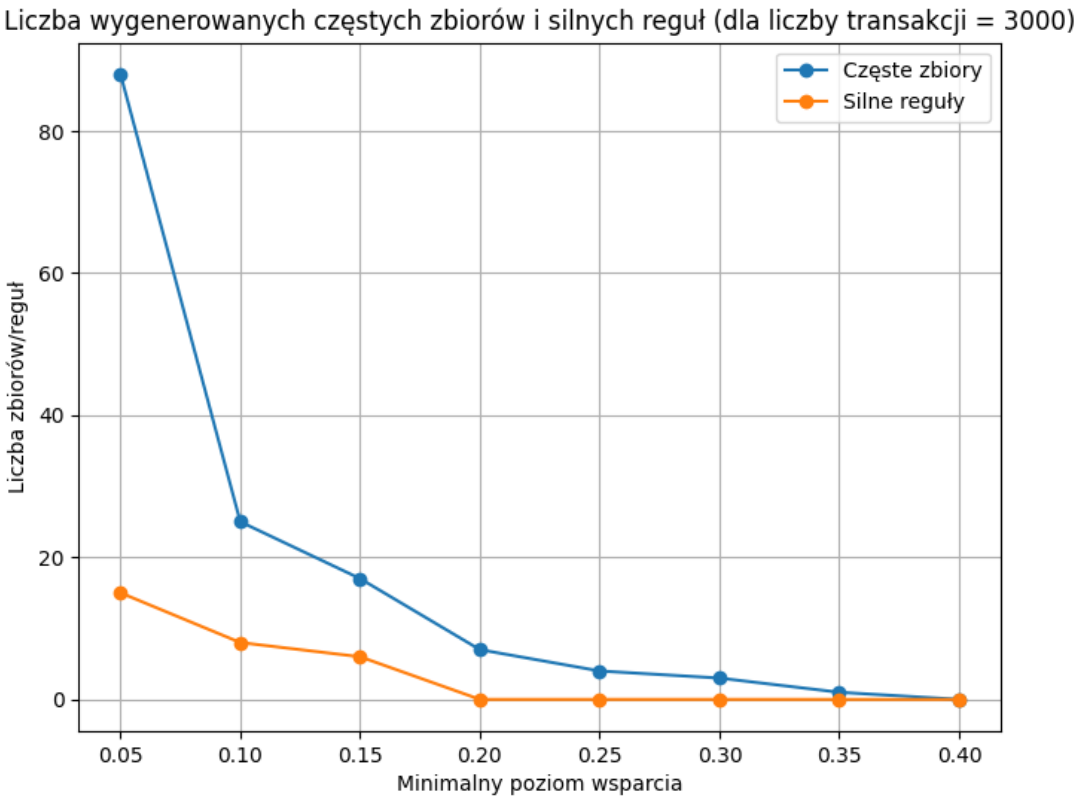
Porównując wykres z rysunku 7 z wykresami z rysunku 5 oraz 6 widać, że wpływ na wykładniczy charakter przebiegów z rysunków 5 oraz 6 ma ustawienie parametru min_sup jako jednakowa wartość dla każdej liczby transakcji. Naturalnym jest, że dla dużej liczby transakcji będzie dużo więcej częstych zbiorów niż dla mniej licznych zbiorów w przypadku gdy dla każdego ze zbiorów mamy jednakową wartość progową. A im więcej częstych zbiorów tym więcej kombinacji jest generowanych i czas działania algorytmu się wydłuża.



Rys. 7: Czas działania algorytmu od wielkości zbioru wejściowego

7.7 Liczba generowanych czestych zbiorów i silnych reguł w zależności od wartości min_sup, przy stałej wielkości zbioru wejściowego

Z rysunku 8 wynika, że wraz ze wzrostem procentowego poziomu min_sup ilość generowanych częstych zbiorów maleje. A co za tym idzie zmniejsza się również liczba generowanych silnych reguł asocjacyjnych.

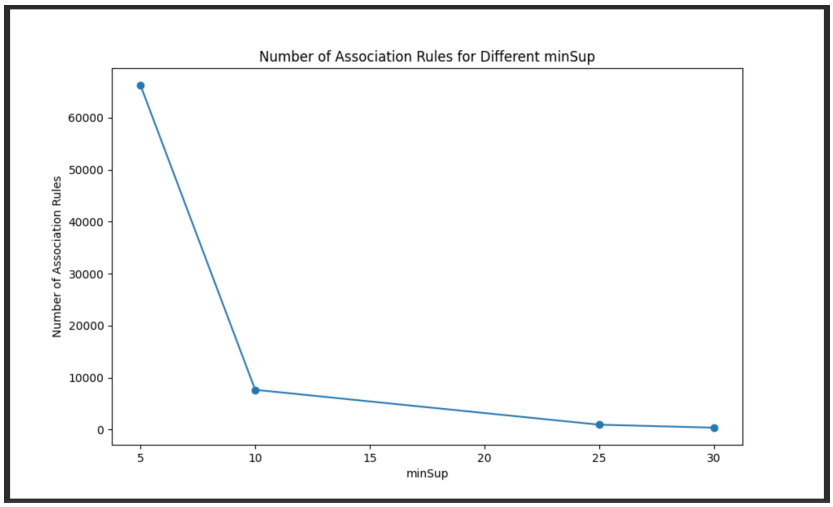


Rys. 8: Liczba generowanych częstych zbiorów i silnych reguł w zależności od min_sup.

8 Eksperymenty, obserwacje i wnioski dla algorytmu fpgrowth.

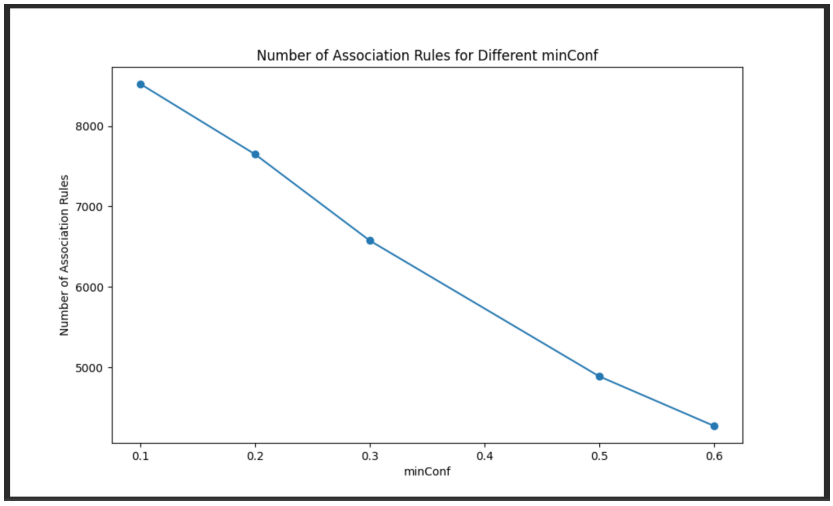
8.1 Zależność od parametrów minSupport i minConf.

Przeanalizowano ilość odkrtyych reguł dla różnych wartośći min support.



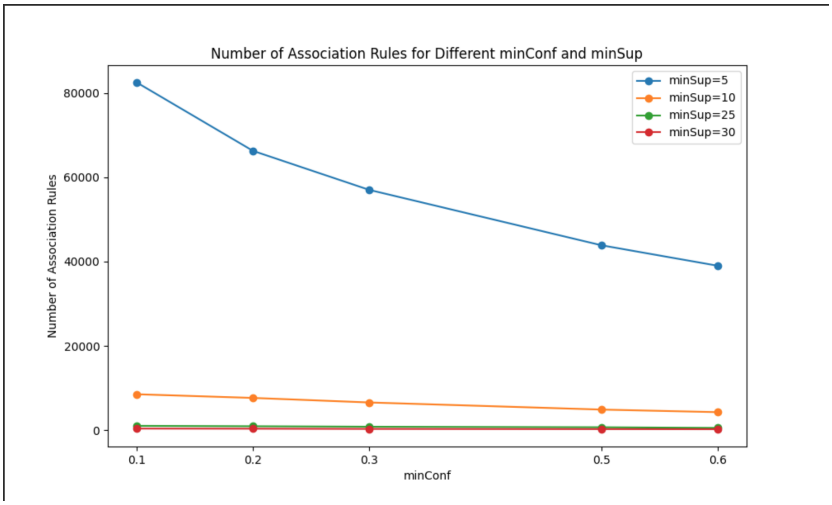
Rys. 9: Ilość odkrytych reguł asocjacyjnych dla różnych wartości minimalnego wpsarcia, minconf = 0.2 i 500 transakcji.

Przeanalizowano ilość odkrtyych reguł dla różnych wartośći min confidence.



Rys. 10: Ilość odkrytych reguł asocjacyjnych dla różnych wartości minimalnego zaufania reguły, minsup = 10 i 500 transakcji.

Poniżej także zestawienie wykresów ze zmiennymi obydwoma parametrami..



Rys. 11: Ilość odkrytych reguł asocjacyjnych dla różnych wartości minialnego wsparcia i minimalnego zaufania reguły dla 500 transakcji.

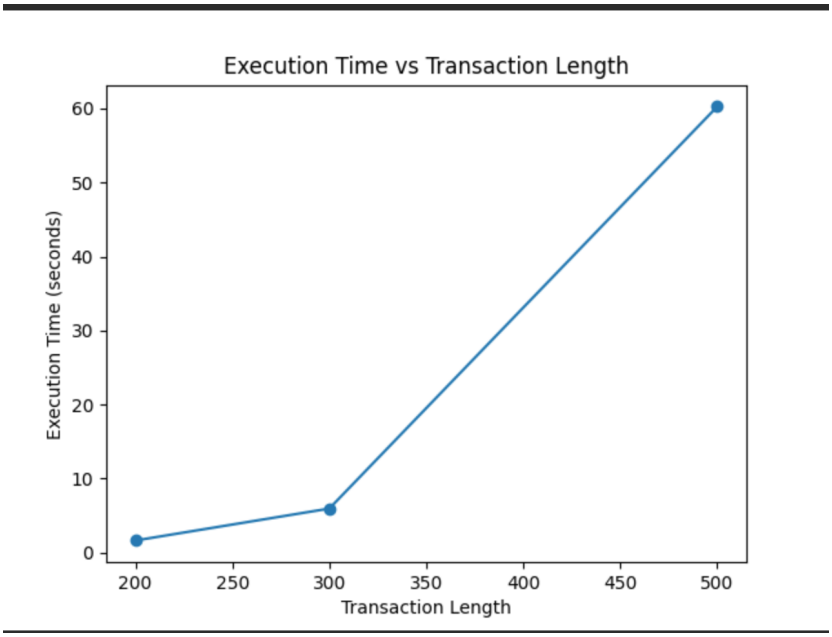
Z wykresów wynika, że minimalne wsparcie ogranicza nieliniowo zbiór generowanych reguł i spada gwałtownie. W przeciwieństwie do minimalnego zaufania, które wyołuje liniową zmianę w ograniczaniu reguł.

8.2 Czas wykonania dla różnej liczby transakcji.

Algorytm fpgrowth najdłużej wykonuje się w fazie odkrywania reguł asocjacyjnych jest to funkcja: associationRule, gdzie przeglądając wszystkie podzbiory iteruje po transakcjach w celu obliczenia wsparcia. Faza budwania drzewa i trawersowania po drzewie jest w porównaniu do czasu wykonywania tej metody pomijalna (przunajmniej dla tych wartości transakcji). Niestety przez to algorytm jest wolniejszy od apriori co należy zoptymalizować, bo jest wbrew teorii.

8.3 Wnioski dla algorytmu fpgrowth.

W algorytmie udało się wykrywać reguły wilepoziomowe, jednak czas działania pozostawia wiele do poprawy. Głównie etap końcowego generowania reguł asocjacyjnych z wszystkich kombinacji zbiorów częstych, mógłby w bardziej efektywny sposób korzystać z wcześniej policzonego wsparcia anie iterować po zbiorze transakcji co daje złożoność O(N2) i jest wąskim gardłem tej implementacji.



Rys. 12: Czas wykonywania w sekundach względem ilości transakcji, dla parametrów minSup = 10, minConf = 0.2.

.

9 Bibliografia

1. Wykład MED 23L, prof. dr hab. Marzena Kryszkiewicz

2. http://smurf.mimuw.edu.pl/external_slides/W4_Wielopoziomowe_i_wielowymiarowe_reguly_asocjacyjne