SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 4851

POSTUPCI ZA UČENJE ASOCIJATIVNIH PRAVILA

Marin Smoljanić

Zagreb, lipanj 2017.

Sadržaj

Uvod

Asocijativna pravila pogodna su za pronalaženje relacijskih odnosa između više varijabli u velikim skupovima podataka. Odnosi između varijabli, takozvana asocijativna pravila, izvode se u obliku implikacija AKO-ONDA.

Važni termini u terminologiji asocijativnih pravila su:

- Element (engl. *Item*)
  - u terminologiji obrade podataka uobičajenije je korištenje par 'atribut-vrijednost'
- Skup elemenata (engl. *Itemset*)
- Transakcija (engl. *Transaction*)
  - korespondira terminu 'primjerak' u području obrade podataka [2]

Većina algoritama asocijativnih pravila najbolje radi nad binarnim ili kategoričkim tipovima podataka što ih čini pogodnima za brzi pregled odnosa između varijabli u velikim skupovima podataka, no pak manje pogodnima za obradu detaljnijih numeričkih skupova podataka.

Među najpoznatije primjere postupaka učenja asocijativnih pravila ubrajamo Eclat, Tertius, OPUS te dva algoritma koji će detaljnije biti pojašnjeni unutar ovog rada, a to su algoritmi Apriori i FP-Growth.

Programska potpora implementacije algoritama Apriori ostvarena je kroz Splunk – sustav upravljanja podatcima.

1. Osnovni koncepti asocijativnih pravila

Primjer transakcija potrošačke košarice dan je tablicom 1.1. Vrijednost ‘Mlijeko’ ili ‘Kruh’ primjeri su elemenata. Primjer jednog skupa elemenata (engl. *itemset*) dan je izrazom ‘{Mlijeko, Kruh, Pivo}’ dok jedan cijeli redak tablice odgovara jednoj transakciji koja je definirana jednoznačnom vrijednošću ID-a te pridruženim skupom elemenata.

ID	ITEMS
1	{Mlijeko, Brašno}
2	{Sok, Kruh, Brašno, Ubrusi}
3	{Mlijeko, Kruh, Pivo, Sok}
4	{Mlijeko, Kruh, Sok, Pelene}

Tablica 1.1: Primjer transakcija potrošačke košarice

Podatci potrošačke košarice mogu biti predstavljeni binarnim formatom kako je prikazano tablicom 1.2. Svaki redak tablice odgovara jednoj transakciji, a svaki stupac tablice odgovara jednom elementu. Sadržaj transakcije definiran je vrijednostima ‘0’ i ‘1’. Vrijednost elementa ‘1’ govori kako transakcija sadrži navedeni element dok vrijednost ‘0’ predstavlja ne pojavljivanje elementa u transakciji.

ID	Mlijeko	Brašno	Kruh	Ubrusi	Pivo	Sok	Pelene
1	1	1	0	0	0	0	0
2	0	1	1	1	0	1	0
3	1	0	1	0	1	1	0
4	1	0	1	0	0	1	1

Tablica 1.2: Binarna reprezentacija transakcija potrošačke košarice

Definirajmo asocijativno pravilo **X** → **Y (**implikacija AKO-ONDA), gdje su **X** i **Y** dva disjunktna skupa za koje vrijedi **X** ∩**Y** = ∅. U ovako definiranom pravilu varijabla **X** se još naziva antecedens, a varijabla **Y** konzekvens. Vrijednost ili snaga asocijativnog pravila mjeri se u terminima signifikantnosti i pouzdanosti.

Neka je $I = \{i_1, i_2, \ldots, i_d\}$ skup svih elemenata određenog skupa podataka te je $T = \{t_1, t_2, \ldots, t_N\}$ skup svih transakcija, vrijedi da svaka transakcija t_i sadrži određeni podskup skupa svim elemenata I. Jedan *itemset* je upravo jedan podskup skupa svih elemenata I, a jedna transakcija može sadržavati više *itemsetova*. *Itemset* koji sadrži *k* elemenata nazivamo *k*-itemset, npr. {Mlijeko, Kruh, Pivo} je jedan primjer *3*-itemseta. [1]

Uvodimo novi pojam, broj signifikantnosti (engl. *support count*). *Support count* predstavlja broj transakcija koje sadrže pojedini itemset. *Support count itemseta* X dan je izrazom:

$$\sigma(X) = |\ \{t_i \mid X \subseteq t_i, t_i \in T\}|\ ,$$

Na primjeru skupa podataka danom tablicom 1.2 možemo primjetiti kako se u transakcijama 3 i 4 pojavljuju elementi 'Mlijeko' i 'Kruh', tj. obje transakcije sadrže *itemset* {Mlijeko, Kruh}, a samim time *support count itemseta* {Mlijeko, Kruh} jednak je 2.

1.1. Signifikantnost

Značaj ili signifikantnost (engl. *Support*) je mjera frekvencije pojavljivanja pojedinog *itemseta* u skupu svih transakcija ili, gledano iz perspektive asocijativnih pravila, mjera primjenjivosti asocijativnog pravila nad skupom danih transakcija. Signifikantnost se računa izrazom:

$$s(X \rightarrow Y) = \sigma(X \cup Y) / N\ ,$$

gdje $\sigma(X \cup Y)$ predstavlja *support count* vrijednost skupa svih elemenata koji sačinjavaju pravilo, a N je ukupan broj transakcija u danom skupu podataka. [1]

Uzmimo, temeljem podataka prikazanih tablicom 1.2, za primjer pravilo {Mlijeko, Kruh} → {Sok}. *Support count* unije svih elemenata pravila {Mlijeko, Kruh, Sok} iznosi σ({Mlijeko, Kruh, Sok}) = 2 (postoje dvije transakcije koje sadrže sva 3 elementa, a to su

transakcije 3 i 4). Kako navedeni skup podataka sadrži ukupno 4 transakcije signifikantnost pravila je $s(\{\text{Mlijeko, Kruh}\} \rightarrow \{\text{Sok}\}) = \sigma(\{\text{Mlijeko, Kruh, Sok}\}) / N = 2/4 = 0.5$

1.2. Pouzdanost

Pouzdanost (engl. *Confidence*) određuje koliko često se elementi iz **Y** skupa pojavljuju u transakcijama koje sadrže skup elemenata **X**. Drugim riječima, pouzdanost pravila iznosa 0.6 nam govori da kada se u transakciji pojavi skup elemenata **Y**, postoji 60% šanse da će ista transakcija sadržavati također i skup elemenata **X**. Pouzdanost se računa izrazom:

$$c(X \rightarrow Y) = \sigma(X \cup Y) / \sigma(X) .$$

Pouzdanost pravila računa se dijeljenjem *support count* vrijednosti unije svih elemenata pravila $\sigma(X \cup Y)$ sa *support count* vrijednošću antecedensa $\sigma(X)$.
Uzmimo za primjer pravilo {Mlijeko, Kruh} → {Pivo}. Pouzdanost danog pravila iznosi 0.5 jer postoji jedna transakcija (transakcija 3) koja sadrži sve elemente pravila, a dvije transakcije sadrže elemente Mlijeko i Kruh koji sačinjavaju antecedens danog pravila.

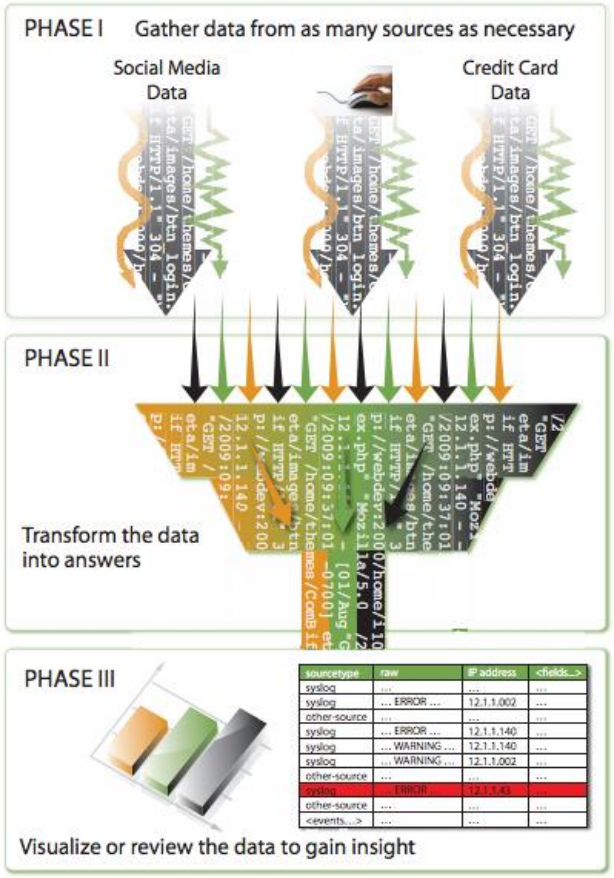
1.3. Minimalan prag signifikantnosti i pouzdanosti

Signifikantnost i pouzdanost su vrlo važne mjere jer generirana pravila s vrlo malom vrijednošću signifikantnosi mogu nastati iz puke slučajnosti. Osim toga, pravila sa vrlo malom vrijednošću signifikantnosti nisu previše korisna u konačnoj formi. Što je veća pouzdanost za dano pravilo X → Y, time sa većom sigurnošću možemo biti sigurni da će konzekvens Y uistinu i biti prisutan u transakcijama koje sadrže antecedens X.

U to ime definiraju se dva nova praga, a to su minimalni prag signifikantnosti i minimalni prag pouzdanosti čije vrijednosti se unaprijed određuju ili se daje korisniku da ih sam odredi.

2. Splunk

Splunk je programski proizvod za centralizirano upravljanje podatcima. Omogućuje jednostavno prikupljanje, pretraživanje i analiziranje podataka bez obzira radi li se o transakcijskim podatcima, dnevničkim zapisima ili bilo kojem drugom tipu podataka bez obzira na količinu istih (od nekoliko megabajta pa da do više od terabajta na dnevnoj razini). U mogućnosti je u stvarnom vremenu prikupiti i indeksirati (o čemu će više biti u nastavku) velike količine podataka iz bilo kojeg IT izvora bez korištenja baze podataka. [6]



Slika 1: Proces obrade podataka kroz Splunk [3]

Dobro svojstvo Splunka jest razdvojenost procesa obrade podataka na tri faze, kao što je prikazano slikom 1. Proces analize podataka jasno je odvojen od procesa prikazivanja analiziranih podataka krajnjem korisniku što je korisno jer se programer može fokusirati na

samu analizu dok mu Splunk omogućava automatsko generiranje korisničkog sučelja, uz manje potrebe vlastitih preinaka automatski generiranih HTML, CSS ili JavaScript dokumenata.

Splunkov sustav za pretraživanje podataka osigurava nadgledanje, uzbunjivanje (pogodno za razvoj tzv. *alerting* ili *monitoring* aplikacija) i izvješćivanje za različite potrebe kao što su upravljanje sigurnosnim događajima ili upravljanje operacijama.

Važni pojmovi, koji će u nastavku biti detaljnije opisani, vezani uz Splunk su:

- Događaj (engl. *Event*)
- Domaćin, izvor i tip izvora (engl. *Host, Source and Source Type*)
- Polja (engl. *Fields*)
- Oznake (engl. *Tags*)
- Indeksi (engl. *Indexes*)
- Pretraga (engl. *Search*)
- Kontrolna ploča (engl. Dashboard)
- Jezik pretrage (engl. Search Processing Language)

2.1. Događaj

Događajem se smatra skup vrijednosti koji ima zabilježeno isto vrijeme ulaza u Splunk aplikaciju. Drugim riječima, event je pojedinačni unos podataka koji može biti predstavljen tekstualnim dokumentom, konfiguracijskim dokumentom, dnevničkim zapisom, transakcijskim skupom podataka…

2.2. Domaćin, Izvor, Tip Izvora

Domaćin je ime fizičkog ili virtualnog uređaja s kojeg dolaze ulazni podatci te je korisno u situaciji kada nas zanimaju svi podatci koji su došli sa točno određenog izvora. Primjerice, Splunk aplikacija može obrađivati log zapise sa većeg broja različitih poslužitelja te u tom slučaju bilježenje imena izvora podataka uvelike olakšava situaciju ako treba grupirati dolazne podatke po izvoru.

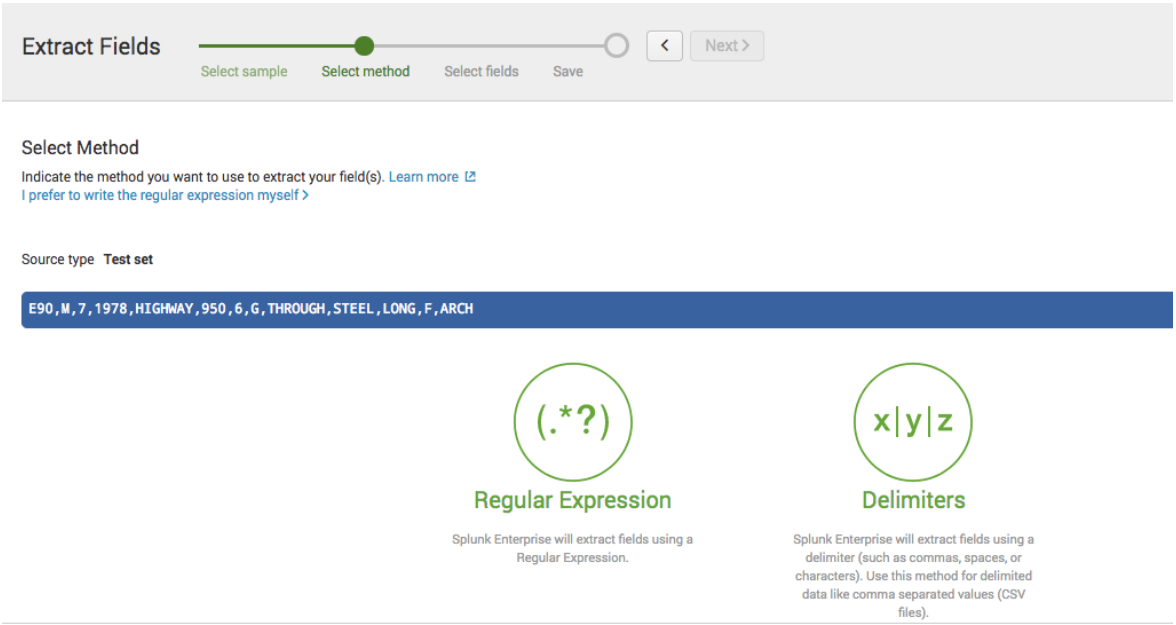
Source ili izvor je ime dokumenta, direktorija ili nekog drugog ulaza s kojeg pristižu određeni eventi. U gornjem primjeru to bi bilo ime dnevničkog zapisa.

Izvori se klasificiraju prema tipu, koji mogu biti neki od poznatih formata dokumenata kao što su recimo csv, tsv, json, xml ili u posebno definiranom tipu podatka od strane krajnjeg korisnika.

2.3. Polja

Polja su parovi atribut-vrijednost koji razdvajaju evente jedne od drugih. Kada bi primjerice obrađivali .csv dokumente Splunk bi automatski prepoznao atribute definirane u prvom retku dokumenta te sve različite vrijednosti (svaki stupac u .csv fileu odgovara jednoj varijabli ili atributu) bi uspješno grupirao prema prepoznatim atributima na temelju regularnih izraza koje primjenjuje na predefinirane tipove podataka.

Ako je potrebno obraditi tip dokumenta koji nije propisan standardnim formatima dokumenata Splunk nudi opciju raščlanjivanja takvih dokumenata pisanjem vlastitih regularnih izraza ili ako se radi o jednostavnijim zapisima moguće je odabrati određeni separator vrijednosti kao što su recimo točke, zarezi ili neki drugi znakovi koje je moguće definirati nakon uspješno učitanog seta podataka (postupak je detaljnije opisan u poglavlju o uputama za korištenje programske potpore) odabiranjem opcije '*Extract fields*' u poslijednjem koraku učitavanja podataka.



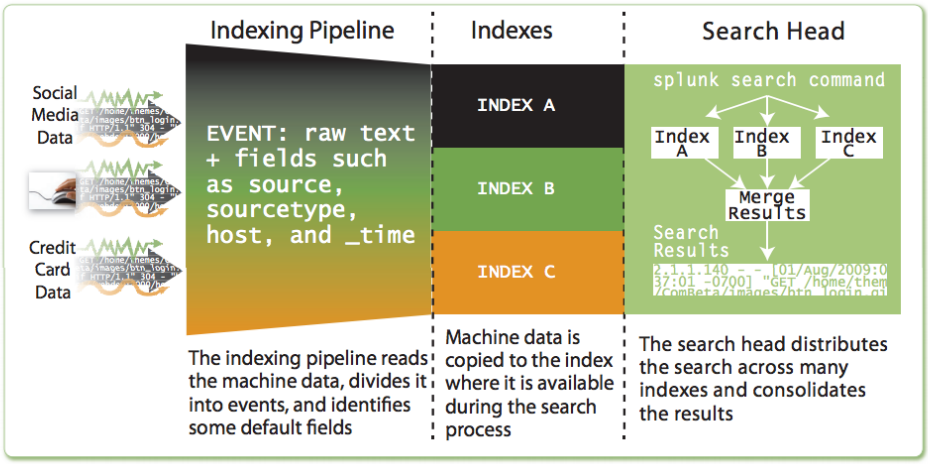
Slika 2: Odabir postupka raščlanjivanja seta ulaznih podataka

2.4. Oznake

Oznaka je vrijednost koja objedinjuje određene skupove vrijednosti polja. Oznakama je moguće grupirati različite parove atribut-vrijednost, domaćine, izvore ili tipove izvora prema nekim zajedničkim obilježjima. Pomoću njih se podatci uspješno mogu organizirati u različite kategorije čime je postignuta dodatna funkcionalnost prije buduće, detaljnije, obrade istih.

2.5. Indeksi

Nakon što su podatci zaprimljeni unutar Splunka, on ih parsira u individualne događaje, određuje vrijeme dolaska podataka, primjenjuje pravila razdvajanja parova atribut-vrijednost i pohranjuje tako obrađene podatke u novo kreirane dokumente koji se nazivaju indeksima. Indeksi postaju novi izvori podatka koje Splunk koristi u budućoj analizi, tj. indeksi postaju zamjena za bazu podataka. [5]



Slika 3: Postupak indeksiranja ulaznih podataka

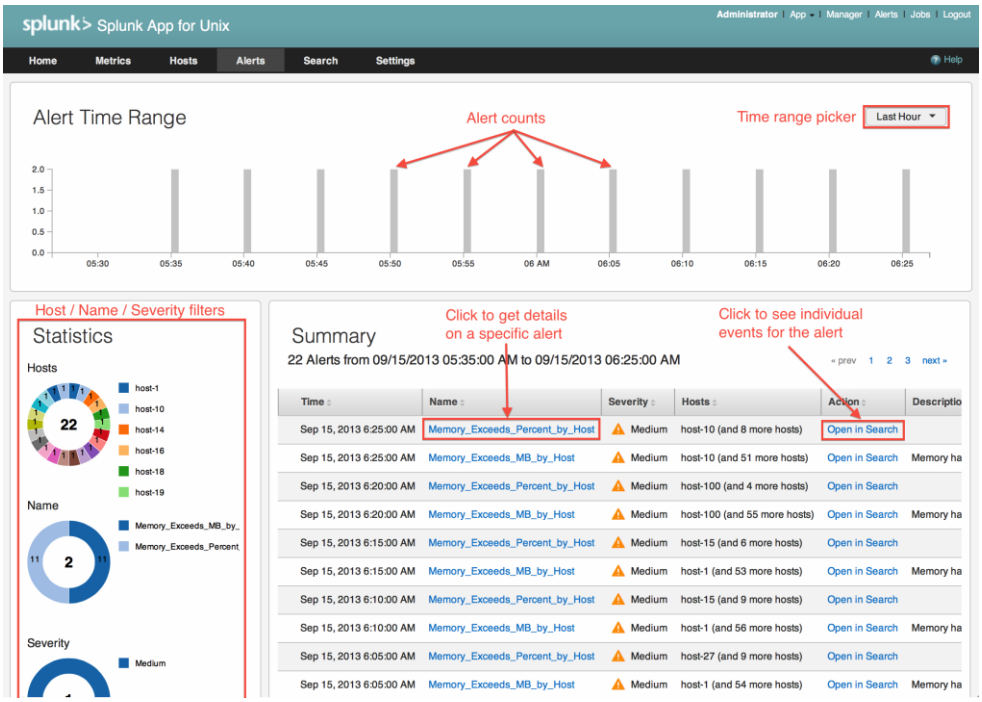
2.6. Pretraga

Pretraga je osnovni način kojim korisnici obrađuju podatke u Splunku. Pomoću pretraga korisnik dohvaća indeksirane podatke, izvršava statističke naredbe nad njima, generira izvješća, traži određene sekvence unutar skupa podataka, identificira uzorke i sl.

Pretrage se pišu pomoću jezika pretraživanja SPL (engl. *Search Process Language*) čime se ulazni skup podataka transformira/reducira/oblikuje prema potrebama u novi, analizom nastali, skup podataka koji predstavlja potrebno rješenje.

2.7. Kontrolna ploča

Prethodno napisane pretrage moguće je spremiti te pomoću njih pogoniti *dashboarde* pomoću kojih se rješenja dobivena od spremljenih pretraga prezentiraju krajnjim korisnicima. *Dashboard* može biti sačinjen od različitih grafikona koji vizualiziraju rješenja dobivena pretragama, panela unutar kojih je moguće dohvatiti nove skupove podataka na kojima se izvršavaju spremljene pretrage ili polja unutar kojih je moguće pisati nove pretrage.



Slika 4 Primjer *dashboarda* unutar Splunka

2.8. Implementacija algoritma unutar Splunka

Splunkov jezik pretraživanja sastavljen je od već prethodno napisanih naredbi, no dani skup naredbi moguće je proširiti vlastitom implementacijom istih. Splunkove naredbe možemo promatrati kao crne kutije koje na ulaz primaju skup podataka nad kojim izvrše potrebne modifikacije te na izlaz daju promijenjeni set podataka. Opisanim principom moguće je implementirati algoritme asocijativnog učenja gdje se algoritam implementira poput naredbe koja prima popis svih transakcija te na izlaz plasira generirana asocijativna pravila ili popis učestalih *itemsetova*, zavisno o potrebi. Vlastite naredbe u Splunku pišu se u programskom jeziku Python u kojem su i implementirani potrebni algoritmi. [1]

3. Algoritam Apriori

Algoritam Apriori 1994. godine predložili su inženjeri računarske znanosti Rakesh Agrawal i Ramakrishnan Srikant. Algoritam je dizajniran kako bi radio nad bazama podataka koje sadrže transakcijske skupove podataka kao što su primjerice sadržaji potrošačkih košarica kupaca u prodavaonicama, bioinformatički transakcijski zapisi ili podatci generirani dubinskom analizom weba.[1]

Apriori, kao i mnogi drugi algoritmi asocijativnih pravila, razdvaja cjelokupni proces obrade podataka na dva potprocesa:

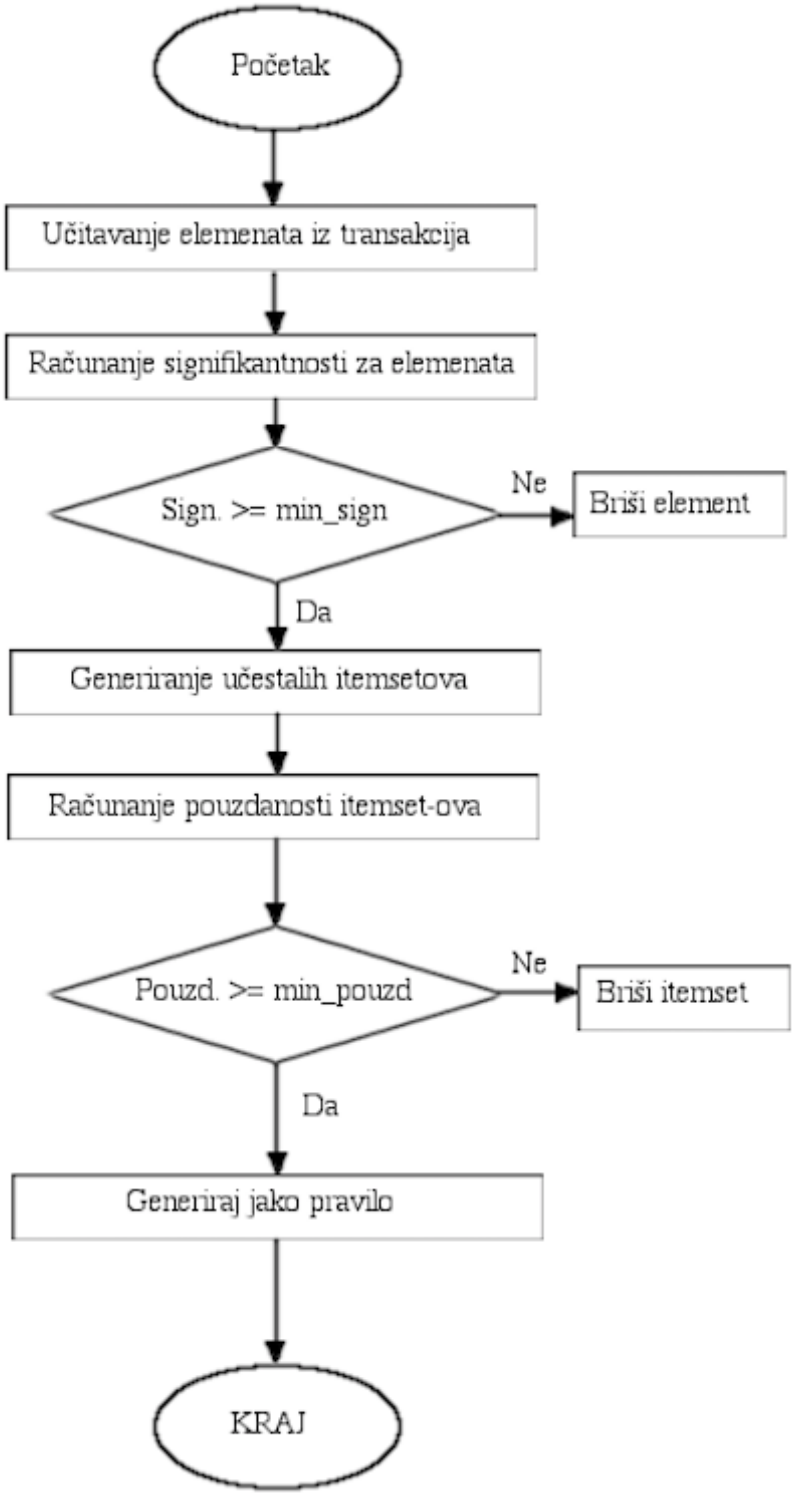
- Generiranje učestalih *itemsetova* (engl. *Frequent itemset generation*)
  - cilj je pronaći sve *itemsetove* koji zadovoljavaju prag minimalne signifikantnosti
  - takvi *itemsetovi* nazivaju se učestalim *itemsetovima* (engl. *frequent itemsets*)

- Generiranje pravila (engl. *Rule generation*)
  - cilj je pronaći sva asocijativna pravila iz prethodno generiranog skupa učestalih *itemsetova* koja zadovoljavaju prag minimalne pouzdanosti
  - tako generirana pravila nazivaju se jakim pravilima (engl. *strong rules*)

Algoritam Apriori na temelju ulaznog skupa podataka generira asocijativna pravila koja zadovoljavaju unaprijed zadane minimalne pragove, prag signifikantnosti te prag pouzdanosti.

Pravila generirana provedbom algoritama asocijativnih pravila trebaju biti razmatrana s dozom opreza. Zaključak donesen na temelju asocijativnog pravila ne povlači nužno i kauzalnost istoga jer pravilo zapravo sugerira samo na jaku povezanost antecedensa i konzekvensa generiranog pravila. Kauzalnost, s druge strane, zahtjeva poznavanja uzročno posljedičnih veza atributa antecedensa i konzekvensa.

Tijek izvođenja algoritma Apriori prikazan je slikom 2.



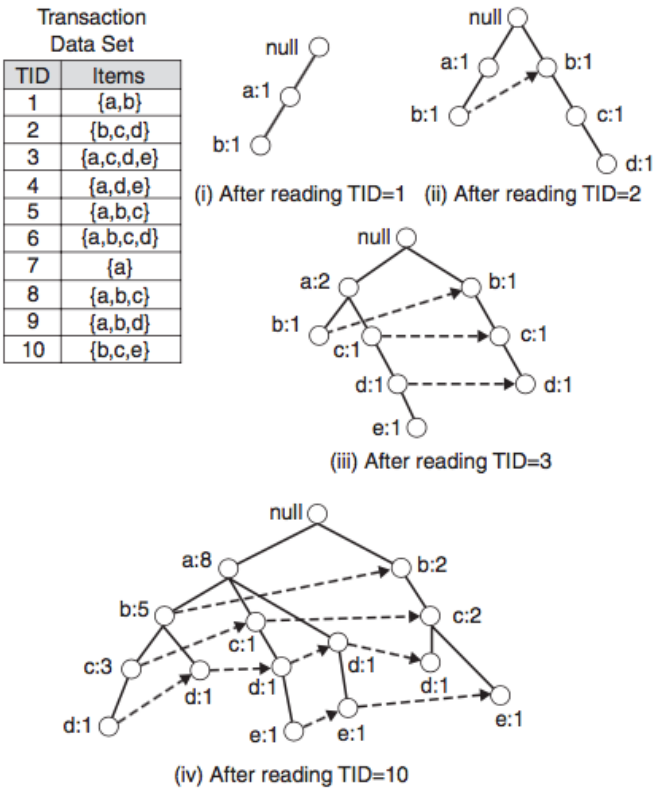
Slika 5: Dijagram tijeka izvođenja algoritma Apriori

4. Algoritam FP-Growth

Algoritam FP-Growth u usporedbi sa algoritmom Apriori primjenjuje potpuno drugačiji način generiranja učestalih *itemsetova*. Šifrira ulazni skup podataka u kompaktnu strukturu FP-stabla iz kojeg onda izvlači učestale *itemsetove*.

FP-stablom algoritam zapravo komprimira ulazni skup podataka čime stablo postaje novi izvor podataka nad kojim algoritam zasniva svoj danji postupak. Stablo se gradi zasebnim preslikavanjem svake transakcije ulaznog skupa podataka na stazu (engl. *path*) koja ga reprezentira. Staze se međusobno mogu prekapati, a što se više staze preklapaju tim se ulazni skup podataka više komprimira (smanjuje se količina ulaznih informacije koje je potrebno obraditi). Ako se veličina FP-stabla može smanjiti dovoljno da ga se pohrani u privremenu memoriju učestale *itemsetove* moguće je izvlačiti izravno iz memorije bez potrebe za dodatnim iteriranjem nad podatcima spremljenim na disku. [1]

Primjer izgradnje FP-Growth stabla dan je u nastavku.



Slika 6: Ilustracija izgradnje stabla FP-Growth

Kao što je vidljivo na slici 5 ulazni skup podataka sastoji se od 10 transakcija te skupa od 5 različitih elemenata. Inicijalno, stablo sadrži jedan korijenski čvor čija je vrijednost null. Iz korijenskog čvora se dalje gradi struktura cijelog stabla. Ilustracija prikazuje prva tri koraka izgradnje stabla te naposlijetku konačan izgled cijelog stabla. Između prve dvije transakcije ulaznog skupa ne postoji preklapanje slijeda elemenata te grana obje transakcije započinje iz korijenskog čvora. Treća transakcija bilježi zajednički prvi element ‘a’ s prvom transakcijom čime dijeli svoj prvi čvor putanje sa prvom transakcijom te daljnje grananje započinje iz zajedničkog čvora ‘a’. Osim različitih čvorova i grana FP-stablo sadrži dodatno i pokazivače koji ulančavaju čvorove jednake vrijednosti.

Algoritam izvršava dva inicijalna prolaza nad ulaznim skupom podataka. U prvom prolazu izračunava *support* vrijednosti pojedinih elemenata (u našem primjeru element ‘a’ je najučestaliji, a redom iza njega slijede elementi ‘b’, ‘c’, ‘d’ pa ‘e’), a drugom iteracijom algoritam izgrađuje stablo gore opisanim postupkom.

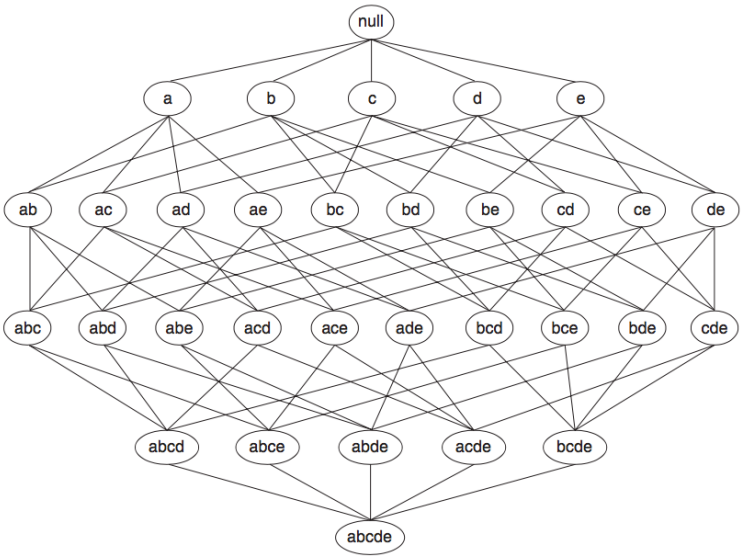
Najbolji slučaj za izgradnju FP-stabla je ulazni skup podatak u kojemu svaka transakcija ima identičan skup elemenata. U tom slučaju stablo bi imalo samo jednu granu koja bi reprezentirala sve transakcija. Najgori slučaj za izgradnju stabla bi bio ulazni skup transakcija od kojih svaka sadrži različit skup elemenata čime bi veličina komprimiranog stabla bila ista kao i ulazni skup podataka. [4]

5. Implementacija algoritma

Kako oba algoritma imaju jednaku podjelu potprocesa (generiranje učestalih *itemsetova* te generiranje pravila), ishod na kraju oba procesa je isti, a razlika je u načinu implementacija procesa generiranja učestalih *itemsetova* dok potproces generiranja pravila može biti zajednički za oba algoritma. Potprocesi generirannja učestalih *itemsetova* te implementacija istih biti će opisana pojedinačno za svaki algoritam, dok će implementacija potprocesa generiranja pravila biti objašnjena zajednički za oba algoritma.

5.1. Generiranje učestalih itemsetova principom Apriori

Radi lakše vizualizacije sve moguće kombinacije *itemsetova* korisno je prikazati mrežnim prikazom vidljivim na slici 5. Svaka razina dane mreže prikazuje skup *k*-itemseta koji brojem odgovaraju pojedinoj razini. Gledajući odozgo prema dolje numeracija razina mreže počinje nulom gdje nam je nulti redak zapravo početno stanje postojećih *itemsetova,* a to je vrijednost null. Prva razina mreže prikazuje moguće kombinacije *1*-itemseta, druga razina prikazuje sve moguće kombinacije *2*-itemseta i tako redom dalje.



Slika 7: Mrežni prikaz *itemsetova*

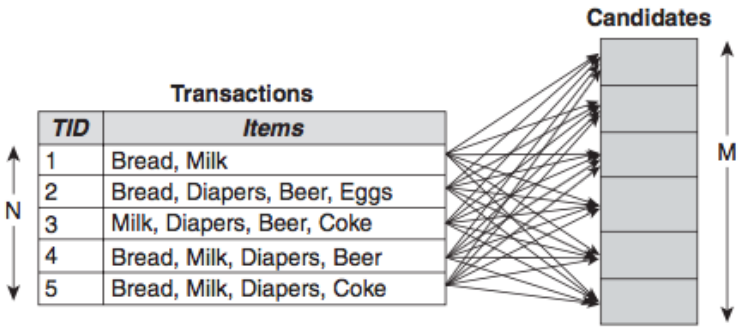
Ako sa K označimo ukupan broj elemenata (u primjeru sa slike 5 to su I={a, b, c, d, e}), a varijabla Y predstavlja razinu mreže koju promatramo ukupan broj kombinacija različitih

itemsetova možemo računati kao X = K povrh Y. Tako, primjerice, vidimo da ukupno postoji 10 mogućih kombinacija *3*-itemseta i *2*-itemseta te po 5 mogućih kombinacija *1*-itemseta i *4*-itemseta.
Općenito, skup podataka koji sadrži K elemenata može potencijalno generirati 2^{K-1} učestalih *itemsetova*, isključujući null *itemset*.
Dva moguća principa generiranja učestalih *itemsetova* biti će opisana u nastavku, a to su princip Apriori te iterirajući (engl. *Brute-force*) princip.

5.1.1. Iterirajući princip generiranja učestalih itemsetova

Itrerirajućim pristupom generiranja učestalih *itemsetova* računamo *support count* vrijednost svakog kandidata (pojedini *itemset* iz skupa svih *itemsetova* mrežnog prikaza naziva se kandidatom jer za vrijeme računanja *support count* vrijednosti svih *itemsetova* svi oni su potencijalni kandidati za uvrštavanje među učestale *itemsetove*, a to ovisi o postavljenim minimalnim pragovima signifikantnosti) mreže *itemsetova*.

Iterirajućim postupkom uspoređujemo svakog kandidata sa svakom transakcijom. Ilustracija postupka prikazana je slikom 6. Ako *iterirajući* po svim transakcijama te svim kandidatima utvrdimo za pojedinog kandidata postojanje unutar transakcije *support count* vrijednost kandidata povećavamo za 1. Navedeni postupak može biti veoma skup (gledano iz perspektive korištenja resursa računala) jer zahtjeva O(NMw) usporedbi, gdje je N broj svih transakcije, $M = 2^{K-1}$ je ukupan broj kandidata, a varijabla w predstavlja najveći broj elemenata koje jedna od transakcija sadrži.



Slika 8: Ilustracija koraka iterirajućeg principa generiranja učestalih *itemsetova*

Opisani postupak moguće je poboljšati smanjivanjem kompleksnosti računa reduciranjem broja ukupnih usporedbi ili reduciranjem ukupnog broja kandidata koje se provodi upravo u principu Apriori generiranja učestalih *itemsetova*.

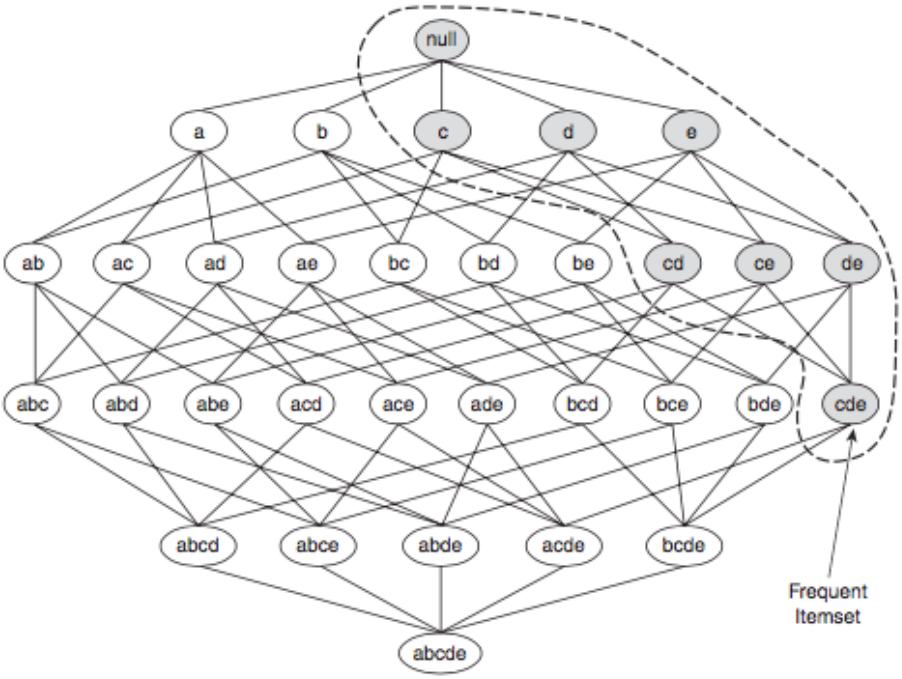
5.1.2. Generiranje učestalih itemsetova principom Apriori

Osnovna ideja principa Apriori jest smanjiti broj potencijalnih kandidata prije računanja njihovih *support count* vrijednosti. Proces smanjivanja broja kandidata se još naziva podrezivanjem. Teorem koji opisuje princip Apriori je sljedeći:

Ako je neki itemset učestal, onda su i svi itemsetovi izvedeni iz njega učestali.

Teorem 1: Princip Apriori

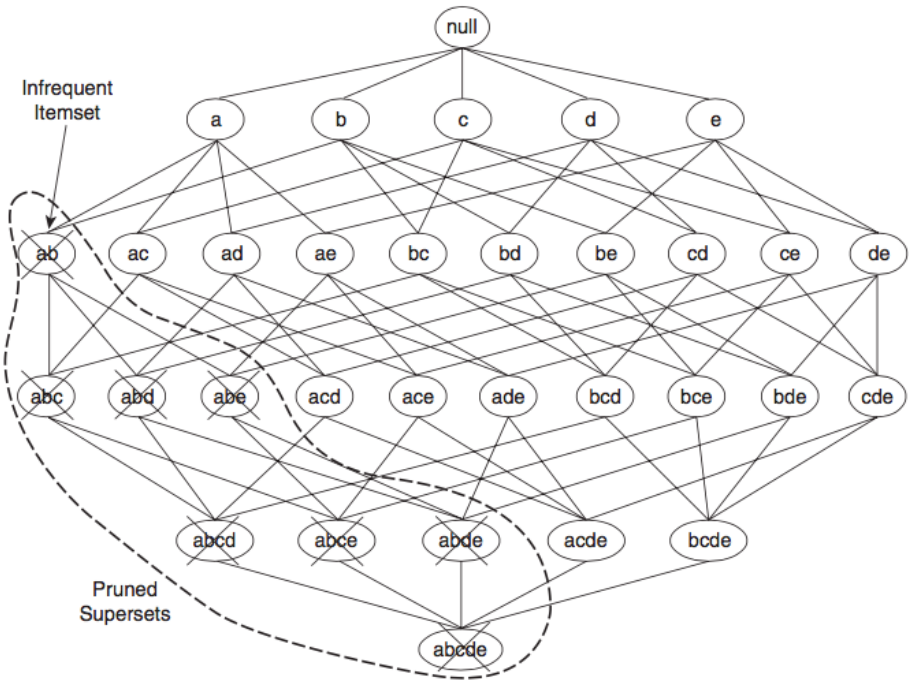
Opisani princip ilustriran je slikom 7. Za primjer je uzet *itemset* It = {c, d, e}. Ako se za njega utvrdi učestalost pojavljivanja onda, prema teoremu 1, možemo bez daljnjeg računa sve *itemsetove* čiji elementi spadaju u podskup skupa elemenata *itemseta* It (*poditemseti itemseta* It) proglasiti učestalima. [1]



Slika 9: Ilustracija principa Apriori

Prethodno opisanim principom Apriori u kojem na temelju učestalosti *itemseta* njegove *poditemsetove* proglašavamo učestalima nismo skratili broj potrebnih usporedbi koje smo morali provesti jer prije nego smo odredili *support cou*nt vrijednost učestalog *itemseta*, u našem primjeru *itemseta* It, morali smo izračunati *support count* vrijednosti njegovih *poditemsetova*.

Posao nam uvelike olakšava činjenica kako *support count* vrijednost nekog *itemseta* neće nikada premašiti *support cou*nt vrijednosti njegovih *poditemsetova*. Prethodna spoznaja još je poznata kao antimonotonost mjere signifikantnosti koja nam zapravo omogućuje primjenu inverznosti principa Apriori. Drugim riječima, ako se za neki *itemset* utvrdi kako nije učestal onda možemo i sve njegove *naditemsetove* proglasiti ne učestalima. Ovime smo uveli novi pojam koji je još poznat kao podrezivanje na temelju signifikantnosti (engl. *support-based pruning*). Ovim postupkom poprilično je smanjen eksponencijalni prostor kandidiranih *itemsetova*.



Slika 10: Ilustracija podrezivanja na temelju signifikantnosti

Primjer generiranja učestalih *itemsetova* dan je slikom 10. Prva tablica prikazuje *1*-itemset skup vrijednosti te izračunate *support count* vrijednosti za iste. Za potrebe ovog primjera

prag minimalne vrijednosti signifikantnosti postavljen je na 60%, što povlači vrijednost 3 kao minimalan iznos *support counta*. Elementi "Cola" i "Eggs" ne zadovoljavaju minimalan *support count* te se *itemsetovi* iduće razine, razine 2, računaju na temelju skupa elemenata {Beer, Bread, Dipers, Milk}. S obzirom da je veličina učestalog *1*-itemset skupa jednaka 4, a potrebno je generirati *2*-itemset skup, broj novo generiranih *2*-itemset skupova iznosi (4 povrh 2) = 6. Prethodno opisanim postupkom utvrđena su dva *itemseta* razine 2 kao neučestala te se isključuju iz danjeg računa skupa *itemseta* iduće razine, razine 3. Kako je broj različitih elemenata skupa *2*-itemseta jednak 3, a potrebno je generirati *3*-itemset skup broj mogućih ishoda je (3 povrh 3) = 1.

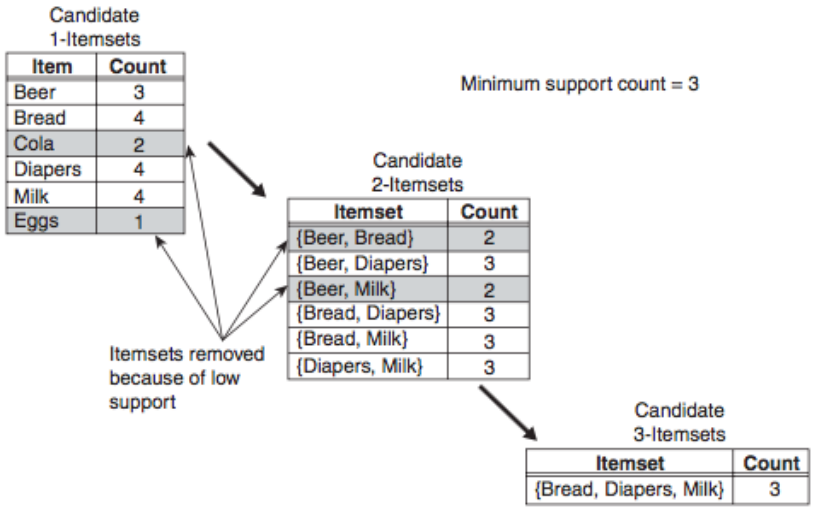


Figure 6.5. Illustration of frequent itemset generation using the *Apriori* algorithm.

Slika 11: Ilustracija generiranja učestalih *itemsetova* principom Apriori

Iterirajuća metoda generiranja učestalih *itemsetova* za isti skup ulaznih podataka generirala bi

$$(6 \text{ povrh } 1) + (6 \text{ povrh } 2) + (6 \text{ povrh } 3) = 6 + 15 + 20 = 41$$

kandidata, dok primjenom principa Apriori broj generiranih kandidata iznosi

$$(6 \text{ povrh } 1) + (4 \text{ povrh } 2) + (3 \text{ povrh } 3) = 6 + 6 + 1 = 13,$$

što odgovara smanjenju broja generiranih kandidata za čak 68%.

Pseudokod implementacije principa Apriori dan je u nastavku.

```
1: fun frequent_itemset_apriori_generation(_items, values):
2:   min_sup = N*SUPP_TRESH, k=1, init_dict _F, init_dict _C
3:   _F[k] = {i|i∈_items ∧ σ({i}) ≥ min_sup}
4:   while True:
5:        k += 1
6:        _C[k] = apriori_gen(_F, k)
7:        for transaction in values:
8:             _C[k] = subset(_C[k], transaction)
9:        for end
10:       _F[k] = {c|c∈_C[k] ∧ σ({c}) ≥ min_sup}
11:       if not _F[k]: break
12:  while end
13:  rule_generation(_F, SUPP_TRESH, _items)
```

Pseudokod 1: Pseudokod generiranje učestalih *itemsetova* principom Apriori

Varijabla `min_sup` predstavlja minimalnu vrijednost *support counta* koja se računa na temelju umnoška ukupnog broja transakcija `N` analiziranog seta podataka te minimalne vrijednosti *supporta* `SUPP_TRESH` koju korisnik proizvoljno odabire. Osim varijable `min_sup` u drugom retku su inicijalizirani varijabla `k` koja označava *k*-itemset razinu za koju tražimo učestale *itemsetove* te varijable *`_F`* i *`_C`* koje su tipa riječnik (engl. *dictionary*), a služe za pohranu buduće izračunatih učestalih *itemsetova* i kandidiranih *itemsetova*. Iteracija algoritma se provodi sve dok za određenu razinu `k` više ne postoji niti jedan učestali *itemset* što se provjerava u liniji 11. Kako algoritam koristi princip Apriori generiranja učestalih *itemsetova*, a ne iterirajući u liniji 6 se poziva metoda *`apriori_gen`* koja vrača listu potencijalnih budućih učestalih *itemsetova* (listu kandidata) koju generira na temelju prethodno izračunatih učestalih (*k-1*)-itemsetova. Iz tog razloga potrebno je prije petlje koja počinje u retku 4 napraviti inicijalni prolaz kroz sve transakcije kojim se računa skup učestalih *1*-itemsetova. Linijama 7 i 8 se vrši iteracija nad svim transakcijama gdje se u svakom koraku poziva metoda *`subset`* koja provjerava postojanje kandidiranih *itemsetova* unutar pojedine transakcije te ukoliko se utvrdi postojanje povećava se *support count* vrijednost kandidata. Dio algoritma koji na temelju izračunatih *support count* vrijednosti kandidata određuje učestale *itemsetove* prikazan je

linijom 10. Nakon izračunatih učestalih *itemsetova* slijedi završni korak algoritma, generiranje asocijativnih pravila, koji je prikazan linijom 13.

5.2. Generiranje asocijativnih pravila

Iz svakog učestalog *k*-itemseta moguće je izgenerirati 2^{K-2} asocijativnih pravila (zanemarujući pravila koja imaju prazan antecedens ili konzekvens). Asocijativno pravilo se generira na temelju elemenata koje sadrži pojedini *itemset*. *Itemset,* npr. W, razdvaja se na dva ne prazna *poditemseta,* U i W-U, takvih da implikacija U → W-U zadovoljava prag minimalne pouzdanosti.

Recimo da je I = {a, b, c} učestali *itemset*. Za dani *itemset* postoji šest kandidirajućih asocijativnih pravila. {a, b} → {c}, {a, c} → {b}, {b, c} → {a}, {a} → {b, c}, {b} → {a, c} i {c} → {a, b}. Računanje pouzdanosti danih pravila ne zahtjeva dodatne operacije računanje nad *itemsetovima* jer se pouzdanost može izračunati na temelju već prethodno izračunatih *support count* vrijednosti za pojedini *itemset*. Primjerice, za pravilo {a, b} → {c}, pouzdanost se računa izrazom σ({a, b, c})/σ({a, b}) (prisjetimo se formule koja glasi $c(X \rightarrow Y) = \sigma(X \cup Y) / \sigma(X)$). [1]

Pseudokod implementacije generiranja pravila dan je u nastavku.

```
1: fun rule_generation(_F, _items):
2:     for f_itemset in _F[k], k>=2:
3:         if len(f_itemset) == 2:
4:             app_genrules_2_itemsets(f_itemset, _F, _items)
5:         else:
6:             H1 = {i|i∈f_itemset}
7:             app_genrules(f_itemset, H1, _F, _items)
8:     end for
```

Kako *itemset* mora minimalno sadržavati 2 elementa (jedan element antecedensa te jedan element konzekvensa) prije samog procesa generiranja pravila potrebno je napraviti

iteraciju kojom se iterira po svim učestalim *k*-itemsetovima gdje je k>=2. Zbog načina implementacije postupka generiranja pravila kao ulazni parametar funkciji `app_genrules` osim *itemseta* `f_itemset` iz kojeg se izvlače pravila, skupa svim učestalih *itemsetova* `_F`, te liste elemenata `_items` potrebno je proslijediti još listu svih potencijalih *1*-itemset konzekvenata čije generiranja je prikazano u liniji 6. Postupak generiranja pravila iz *2*-itemsetova zbog određenih programskih ograničenja programskog jezika Python bilo je potrebno odvojiti od postupka generiranja pravila iz *k*-itemsetova gdje je k>=3 (linije 3 i 4).

```
1: fun app_genrules(f_itemset, Hm, _F, _items):
2:    k = len(f_itemset)
3:    m = len(Hm)
4:    if k>m+1:
5:       Hm+1 = apriori_gen(f_itemset, (m+1))
6:       for each consequent_c in Hm+1:
7:          conf = σ(f_itemset)/σ(f_itemset – consequent_c)
8:          if conf >= min_conf:
9:               output the rule (f_itemset – consequent_c)
                 → consequent_c
10:         else:
11:              delete consequent_c from Hm+1
12:         end if
13:      app_genrules(f_itemset, Hm+1, consequent, _F, _items)
```

Funkcija `app_genrules` generira asocijativna pravila na temelju učestalih *itemsetova*. U prva 2 retka definirane su varijable `k` i `m`, gdje `k` predstavlja veličinu *k*-itemseta, a `m` predstavlja broj elemenata potencijalnog konzekventa. Funkcija `app_genrules` je, također, rekurzivna funkcija koja sa iteracijama nastavlja ukoliko je ispunjen uvjet `k>m+1`. Iterira se po svim potencijalnim konzekventima H_{m+1} (linija 6), koji se u funkciju primaju kao argumenti, te se za svakog konzekventa računa pouzdanost kao omjer *support counta* svih elemenata učestalog *itemseta* kojem potencijalni konzekvent pripada i *support count*

vrijednosti svih elemanta *itemseta* bez elemanta od kojeg je sačinjen potencijalni konzekvent. Ako izračunata vrijednost pouzdanosti zadovoljava minimalan prag pouzdanosti `min_conf` ispisuje se pravilo `(f_itemset – consequent_c)` → `consequent_c` inače se potencijalni konzekvent briše iz liste svih potencijalnih konzekvenata H_{m+1}. Linijom 13 prikazan je rekurzivni poziv funkcije `app_genrules`.

Generiranja pravila *2*-itemseta funkcijom `app_genrules_2_itemsets` jednostavnije je od prethodno opisanog postupka za *k*-itemsete. Računaju se dvije vrijednosti pouzdanosti gdje se prva vrijednost pouzdanosti odnosi na pravilo gdje prvi element *itemseta* predstavlja konzekvens i druga vrijednost pouzdanosti na slučaj gdje drugi element *itemseta* predstavlja konzekvens. Ako izračunate vrijednosti zadovoljavaju prag minimalne pouzdanosti ispisuju se pravila oblika:

`prvi_element_itemset` → `drugi_element_itemseta,`

`drugi_element_itemseta` → `prvi_element_itemseta.`

6. Parsiranje ulaznih skupova podataka

Prije konačne provedbe algoritma pojavio se jedan manji problem. Naime, većina skupova podataka nad kojima je moguće izvrtiti proces generiranja asocijativnih pravila ne dolaze formatirani u binarnom zapisu vrijednosti varijabli kako je prikazano tablicom 2 i nad kakvim formatom zapisa vrijednosti zapravo i radi prethodno opisana implementacija principa Apriori. Na *UC Irvine Machine Learning* repozitoriju ili u postojećim skupovima podataka koji dolaze s instalacijom Weke podatci su većinom formatirani kategorički tako da svaki element linije seta podataka predstavlja vrijednost varijable koju reprezentira, a ne logičku vrijednost postojanja ili ne postojanja iste.

Postupak modificiranja ulaznih podataka moguće je implementirati kroz Splunk kao zasebnu naredbu, no ispostavilo se funkcionalnijim napisati ga kao zasebnu skriptu u Pythonu koja će raščlaniti potreban skup podataka prije nego se isti učita u Splunk.

7. Usporedba algoritma Apriori s postojećim rješenjima

Rezultati implementacije algoritma Apriori biti će uspoređeni s već postojećom implementacijom istog u alatu imenom Weka. Weka je programski proizvod koji je kreiran s ciljem rješavanja problema rudarenja podataka (engl. *data mining problems*). Ispitivanje implementiranog algoritma Apriori bit će provedeno na više skupova podataka koji će potom biti analizirani kroz Weku na temelju čijih rezultata će biti napravljene usporedbe.

7.1. Testni skup podataka

Prilikom rada na implementaciji algoritma programskom skriptom u Pythonu generiran je jedan testni pojednostavljeni skup podataka potrošačke košarice u kojem postoji 10 različitih elemenata {BREAD, MILK, SALT, CHOCOLATE, BEER, JUICE, BANANAS, ORANGE, ICECREAM, EGGS}. Svaki redak datoteke predstavlja jednu transakciju u kojoj je sa ‘1’ označeno postojanje proizvoda u transakciji dok ‘0’ predstavlja odsutnost proizvoda u danoj transakciji.

BREAD,MILK,SALT,CHOCOLATE,BEER,JUICE,BANANAS,ORANGE,ICECREAM,EGGS

1,1,1,0,1,0,1,0,0,0
1,1,0,1,1,0,0,1,1,0
0,1,1,0,1,1,1,0,0,1
0,1,1,1,0,1,0,1,0,1
…

Skup podataka 1: Primjer testnog skupa podataka

Podešeni pragovi:

- Minimalan prag signifikantnosti (engl. *Minimum support*) = 0.14
- Minimalan prag pouzdanosti (engl. *Minimum confidence*) = 0.5

Ukupan broj transakcija danog seta podataka iznosi 1000.

K-itemsets counter

ITEMSET_COUNT	K_ITEMSET
10	1
45	2
13	3

Slika 12: Broj generiranih *k*-itemseta na testnom skupom generiranih u Splunku

```
Apriori
=======

Minimum support: 0.14 (140 instances)
Minimum metric <confidence>: 0.5
Number of cycles performed: 18

Generated sets of large itemsets:

Size of set of large itemsets L(1): 10

Size of set of large itemsets L(2): 45

Size of set of large itemsets L(3): 13
```

Slika 13: Broj generiranih *k*-itemseta na testnom skupu analiziranih kroz Weku

Vidljivo je kako je u oba slučaja generiran jednak broj *itemseta* za svaku razinu. U nastavku je dan prikaz generiranih *3*-itemseta uz izračunate *support count* vrijednosti za svaki *itemset* gdje se, također, rezultati vlastite implementacije podudaraju sa Wekinim rezultatima.

Frequent itemset generated

ITEMSET	SUPPORT_COUNT ^
{ BEER, EGGS, MILK }	140
{ BANANAS, EGGS, SALT }	140
{ BEER, ICECREAM, MILK }	140
{ BEER, CHOCOLATE, ICECREAM }	140
{ BREAD, EGGS, ICECREAM }	141
{ BANANAS, ICECREAM, SALT }	141
{ BANANAS, EGGS, ICECREAM }	142
{ BANANAS, EGGS, ORANGE }	143
{ BANANAS, ICECREAM, ORANGE }	144
{ BANANAS, ORANGE, SALT }	146
{ BANANAS, BREAD, SALT }	146
{ BREAD, ICECREAM, SALT }	149
{ BANANAS, BREAD, ICECREAM }	150

Slika 14: 3-itemseti generirani nad testnim skupom u Splunku

```
Large Itemsets L(3):
bread=t salt=t banannas=t 146
bread=t salt=t icecream=t 149
bread=t banannas=t icecream=t 150
bread=t icecream=t eggs=t 141
milk=t beer=t icecream=t 140
milk=t beer=t eggs=t 140
salt=t banannas=t orange=t 146
salt=t banannas=t icecream=t 141
salt=t banannas=t eggs=t 140
chocolate=t beer=t icecream=t 140
banannas=t orange=t icecream=t 144
banannas=t orange=t eggs=t 143
banannas=t icecream=t eggs=t 142
```

Slika 15: 3-itemseti generirani nad testnim skupom u Weki

Manje razlike u rješenjima pojavile su se prilikom generiranja pravila. Vlastitom implementacijom algoritma generirano je 49 asocijativnih pravila dok je Weka izgenerirala 60. Generirana, zajednička, pravila dijele isti iznos pouzdanosti (c(BREAD → ICECREAM) = 0.57, c(MILK → EGGS) = 0.55 ili c(CHOCOLATE → ICECREAM) = 0.55).

Association rules generated

1_ANTECEDENS	2_CONSEQUENS	3_CONFIDENCE
BREAD	ICECREAM	0.570850202429
MILK	EGGS	0.546808510638
CHOCOLATE	ICECREAM	0.546777546778
SALT	BANANAS	0.541747572816
MILK	BEER	0.53829787234
BANANAS	SALT	0.536538461538
BREAD	SALT	0.536437246964
MILK	ICECREAM	0.536170212766
BANANAS	ICECREAM	0.534615384615
ICECREAM	BREAD	0.532075471698

Slika 16: Prvih 10 pravila testnog skupa generirana vlastitom implementacijom algoritma

```
Best rules found:

 1. bread=t banannas=t 260 ==> icecream=t 150    <conf:(0.58)>
 2. bread=t 495 ==> icecream=t 282    <conf:(0.57)> lift:(1.07
 3. salt=t icecream=t 262 ==> bread=t 149    <conf:(0.57)> lif
 4. bread=t eggs=t 248 ==> icecream=t 141    <conf:(0.57)> lif
 5. chocolate=t beer=t 248 ==> icecream=t 140    <conf:(0.56)>
 6. bread=t salt=t 265 ==> icecream=t 149    <conf:(0.56)> lif
 7. bread=t banannas=t 260 ==> salt=t 146    <conf:(0.56)> lif
 8. milk=t icecream=t 252 ==> beer=t 140    <conf:(0.56)> lift
 9. orange=t eggs=t 258 ==> banannas=t 143    <conf:(0.55)> li
10. milk=t beer=t 253 ==> icecream=t 140    <conf:(0.55)> lift
11. milk=t beer=t 253 ==> eggs=t 140    <conf:(0.55)> lift:(1.
12. bread=t salt=t 265 ==> banannas=t 146    <conf:(0.55)> lif
13. orange=t icecream=t 263 ==> banannas=t 144    <conf:(0.55)
14. chocolate=t 481 ==> icecream=t 263    <conf:(0.55)> lift:(
15. milk=t 471 ==> eggs=t 257    <conf:(0.55)> lift:(1.08) lev
16. salt=t orange=t 268 ==> banannas=t 146    <conf:(0.54)> li
```

Slika 17: Prvih 15 pravila testnog skupa generirana u Weki

7.2. Zapisi kongresnog izglasavanja u SAD-u 1984

Ovaj skup podataka sadrži odgovore svakog člana kongresa SAD-a na 16 pitanja koja su provedena 1984. godine kroz anketu. Značenja pitanja skraćena su i prikazana poput varijabli na koje je moguće bilo odgovoriti sa ‘da’ ili ‘ne’. Skup podataka se sastoji od 16 pitanja uz dodatnu varijablu koja predstavlja stranku kojoj član pripada (demokrat ili republikanac) što čini set od 17 diskretnih kategoričkih varijabli. Kako se radi o diskretnim

kategoričkim podatcima bilo ih je potrebno pretvoriti u skup binarnih kategoričkih podataka postupkom opisanim u poglavlju 6. Skup podataka sadrži ukupno 435 primjeraka.

Podešeni pragovi:

- Minimalan prag signifikantnosti = 0.5
- Minimalan prag pouzdanosti = 0.5

K-itemsets counter

ITEMSET_COUNT	K_ITEMSET
12	1
4	2
1	3

Frequent itemset generated

ITEMSET	SUPPORT_COUNT
{ Class_democrat, adoption_of_the_budget_resolution_y, physician_fee_freeze_n }	219
{ adoption_of_the_budget_resolution_y, physician_fee_freeze_n }	219
{ Class_democrat, adoption_of_the_budget_resolution_y }	231
{ Class_democrat, physician_fee_freeze_n }	245
{ Class_democrat, aid_to_nicaraguan_contras_y }	218

Slika 18: 3-itemseti i 4-itemseti generirani vlastitom implementacijom algoritma

Pregledom slike 17 i 16 vidljivo je kako su u oba slučaja generirani jednaki skupovi učestalih *itemsetova*. Kao i na skupu testnih podataka postoji manje odstupanje u generiranim pravilima. Weka je izgenerirala 14 asocijativnih pravila dok je 11 pravila generirano u Splunku. Pravila koja se nalaze u oba skupa generiranih pravila dijele jednake vrijednosti pouzdanosti.

```
Size of set of large itemsets L(2): 4

Large Itemsets L(2):
adoption-of-the-budget-resolution=y physician-fee-freeze=n 219
adoption-of-the-budget-resolution=y Class=democrat 231
physician-fee-freeze=n Class=democrat 245
aid-to-nicaraguan-contras=y Class=democrat 218

Size of set of large itemsets L(3): 1

Large Itemsets L(3):
adoption-of-the-budget-resolution=y physician-fee-freeze=n Class=democrat 219

Best rules found:

 1. adoption-of-the-budget-resolution=y physician-fee-freeze=n 219 ==> Class=democrat 219
 2. physician-fee-freeze=n 247 ==> Class=democrat 245    <conf:(0.99)> lift:(1.62) lev:(0.21)
 3. adoption-of-the-budget-resolution=y Class=democrat 231 ==> physician-fee-freeze=n 219
 4. Class=democrat 267 ==> physician-fee-freeze=n 245    <conf:(0.92)> lift:(1.62) lev:(0.21)
 5. adoption-of-the-budget-resolution=y 253 ==> Class=democrat 231    <conf:(0.91)> lift:(1.4
 6. aid-to-nicaraguan-contras=y 242 ==> Class=democrat 218    <conf:(0.9)> lift:(1.47) lev:(0
 7. physician-fee-freeze=n Class=democrat 245 ==> adoption-of-the-budget-resolution=y 219
 8. physician-fee-freeze=n 247 ==> adoption-of-the-budget-resolution=y 219    <conf:(0.89)> l
 9. physician-fee-freeze=n 247 ==> adoption-of-the-budget-resolution=y Class=democrat 219
10. adoption-of-the-budget-resolution=y 253 ==> physician-fee-freeze=n 219    <conf:(0.87)> l
11. adoption-of-the-budget-resolution=y 253 ==> physician-fee-freeze=n Class=democrat 219
12. Class=democrat 267 ==> adoption-of-the-budget-resolution=y 231    <conf:(0.87)> lift:(1.4
13. Class=democrat 267 ==> adoption-of-the-budget-resolution=y physician-fee-freeze=n 219
14. Class=democrat 267 ==> aid-to-nicaraguan-contras=y 218    <conf:(0.82)> lift:(1.47) lev:(
```

Slika 19: Asocijativna pravila, *3*-itemseti i *4*-itemseti generirani kroz Weku

Rules stats

RULES_EXTRACTED
11

Association rule generated

1_ANTECEDENS	2_CONSEQUENS	3_CONFIDENCE
physician_fee_freeze_n	Class_democrat	0.991902834008
Class_democrat	physician_fee_freeze_n	0.917602996255
adoption_of_the_budget_resolution_y	Class_democrat	0.913043478261
aid_to_nicaraguan_contras_y	Class_democrat	0.900826446281
physician_fee_freeze_n	adoption_of_the_budget_resolution_y	0.886639676113
Class_democrat, adoption_of_the_budget_resolution_y	physician_fee_freeze_n	0.886639676113
adoption_of_the_budget_resolution_y	physician_fee_freeze_n	0.865612648221
Class_democrat, physician_fee_freeze_n	adoption_of_the_budget_resolution_y	0.865612648221
Class_democrat	adoption_of_the_budget_resolution_y	0.865168539326
adoption_of_the_budget_resolution_y, physician_fee_freeze_n	Class_democrat	0.820224719101
Class_democrat	aid_to_nicaraguan_contras_y	0.816479400749

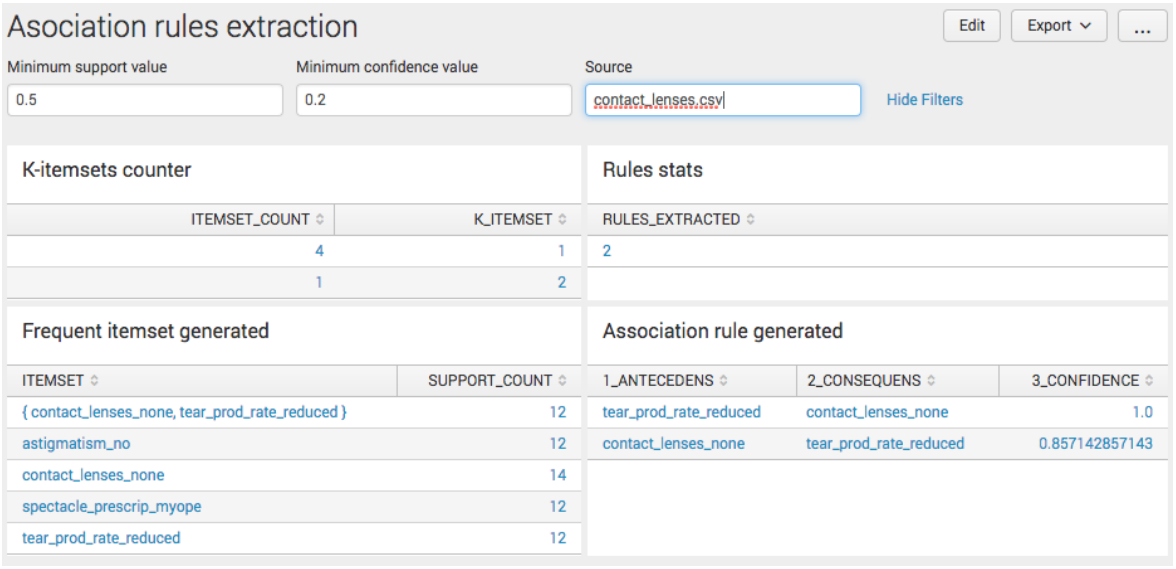
Slika 20: Asocijativna pravila generirana vlastitom implementacijom algoritma

7.3. Zapisi o kontaktnim lećama

U ovom skupu podataka sadržane su informacije o podesivosti kontaktnih leća uz različite parametre koji opisuju pojedinu osobu. Asocijativnim pravilima bi se trebalo utvrditi kakav tip kontaktne leće bi odgovarao osobi koja recimo ima astigmatizam i reproducira dosta suza tijekom dana ili kakav tip leća bi odgovarao mladoj osobi. Ulazi podatci su diskretnog kategoričkog tipa zbog čega ih je trebalo pretvoriti u binarni kategorički. S obzirom da skup podataka sadrži 24 različita primjerka generirano je ukupno 2 pravila koja ukazuju na to kako osoba koja ima suzne oči ne bi trebala nositi leće.

Podešeni pragovi:

- Minimalan prag signifikantnosti = 0.5
- Minimalan prag pouzdanosti = 0.2



Slika 21: Rezultat analize prevedene kroz Splunk nad zapisima o kontaktnim lećama

Možemo primijetiti kako postoji odstupanje u generiranim učestalim *itemsetovima* no razlika nije utjecala na generiranje asocijativnih pravila koja oba imaju jednak iznos pouzdanosti.

```
Apriori
=======

Minimum support: 0.5 (12 instances)
Minimum metric <confidence>: 0.2
Number of cycles performed: 10

Generated sets of large itemsets:

Size of set of large itemsets L(1): 7

Large Itemsets L(1):
spectacle-prescrip=myope 12
spectacle-prescrip=hypermetrope 12
astigmatism=no 12
astigmatism=yes 12
tear-prod-rate=reduced 12
tear-prod-rate=normal 12
contact-lenses=none 15

Size of set of large itemsets L(2): 1

Large Itemsets L(2):
tear-prod-rate=reduced contact-lenses=none 12

Best rules found:

 1. tear-prod-rate=reduced 12 ==> contact-lenses=none 12    <conf:(1)> l
 2. contact-lenses=none 15 ==> tear-prod-rate=reduced 12    <conf:(0.8)>
```

Slika 22: Rezultati analize prevedene kroz Weku

7.4. Zapisi stanja vremena

Zapisi o stanju vremena sadrže informacije koje opisuju pojedini dan koji može biti primjerice kišovit, hladan i vjetrovit. Na temelju informacija o danu izvodi se zaključak o pogodnosti vremena za igranje djece na otvorenome području. S obzirom da dani skup podataka sadrži samo 14 različitih transakcija pragovi su postavljeni poprilično nisko no ipak možemo vidjeti kako su generirana pravila poprilično velike pouzdanosti.

Podešeni pragovi:

- Minimalan prag signifikantnosti = 0.4
- Minimalan prag pouzdanosti = 0.2

K-itemsets counter

ITEMSET_COUNT	K_ITEMSET
4	1
2	2

Frequent itemset generated

ITEMSET	SUPPORT_COUNT
{ humidity_normal, play_yes }	6
{ play_yes, windy_FALSE }	6
humidity_high	6
humidity_normal	7
play_yes	9
windy_FALSE	8

Slika 23: Učestali *itemsetovi* generirani u Splunku

Rules stats

RULES_EXTRACTED
4

Association rule generated

1_ANTECEDENS	2_CONSEQUENS	3_CONFIDENCE
humidity_normal	play_yes	0.857142857143
windy_FALSE	play_yes	0.75
play_yes	humidity_normal	0.666666666667
play_yes	windy_FALSE	0.666666666667

Slika 24: Pravila generirana kroz Splunk

I u ovom primjeru postoji manje odstupanje u generiranim učestalim *itemsetovima* no i u ovom slučaju to nije utjecalo na generirana pravila koja su u oba slučaja jednakih iznosa pouzdanosti.

```
Apriori
=======

Minimum support: 0.4 (6 instances)
Minimum metric <confidence>: 0.2
Number of cycles performed: 12

Generated sets of large itemsets:

Size of set of large itemsets L(1): 6

Large Itemsets L(1):
temperature=mild 6
humidity=high 7
humidity=normal 7
windy=TRUE 6
windy=FALSE 8
play=yes 9

Size of set of large itemsets L(2): 2

Large Itemsets L(2):
humidity=normal play=yes 6
windy=FALSE play=yes 6

Best rules found:

 1. humidity=normal 7 ==> play=yes 6    <conf:(0.86)> l
 2. windy=FALSE 8 ==> play=yes 6    <conf:(0.75)> lift:
 3. play=yes 9 ==> humidity=normal 6    <conf:(0.67)> l
 4. play=yes 9 ==> windy=FALSE 6    <conf:(0.67)> lift:
```

Slika 25: Učestali *itemsetovi* te pravila generirana u Weki

7.5. Zapisi potrošačke košarice

U ovom primjeru kao ulazni set podataka uzeti su zapisi potrošačke košarice jedne prodavaonice. Radi se o binarnim kategoričkim podatcima gdje kao i u testnom primjeru pojedina transakcija sadrži informaciju o postojanju ili ne postojanju pojedinog proizvoda u popisu kupljenih proizvoda. Jedna varijabla je bila diskretnog tipa te je samo nju trebalo pretvoriti u binarni format zapisa.

Set podataka ukupno sadrži 781 transakciju te 219 različitih varijabli.

Podešeni pragovi:

- Minimalan prag signifikantnosti = 0.5
- Minimalan prag pouzdanosti = 0.7

K-itemsets counter

ITEMSET_COUNT	K_ITEMSET
9	1
4	2

Frequent itemset generated

ITEMSET	SUPPORT_COUNT
{ bread and cake, vegetables }	395
{ bread and cake, milk_cream }	398
{ biscuits, bread and cake }	410
{ bread and cake, fruit }	408
baking needs	465
biscuits	504
bread and cake	570
frozen foods	470
fruit	518
juice_sat_cord_ms	430
margarine	396
milk_cream	474
vegetables	501

Slika 26: Učestali *itemsetovi* generirani kroz Splunk

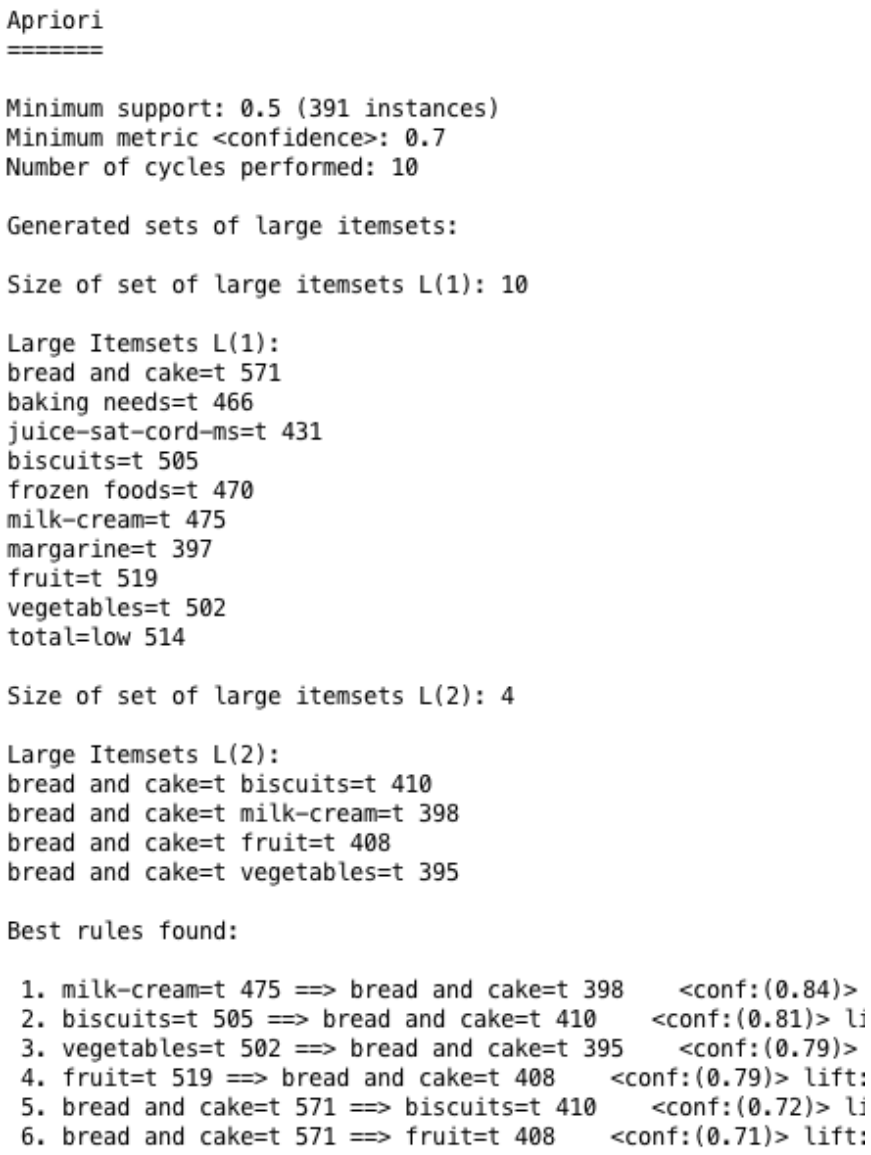
Rules stats

RULES_EXTRACTED
6

Association rule generated

1_ANTECEDENS	2_CONSEQUENS	3_CONFIDENCE
milk_cream	bread and cake	0.839662447257
biscuits	bread and cake	0.813492063492
vegetables	bread and cake	0.788423153693
fruit	bread and cake	0.787644787645
bread and cake	biscuits	0.719298245614
bread and cake	fruit	0.715789473684

Slika 27: Asocijativna pravila generirana kroz Splunk



Slika 28: Učestali *itemsetovi* i asocijativna pravila generirana u Weki

Weka je generirala jedan učestali *itemset* više (total=low 514) te joj ostali učestali *itemsetovi* imaju vrijednost *support counta* za 1 veću od učestalih *itemsetova* generiranih kroz Splunk (osim ‘frozen_foods' koji je u oba slučaji generiran uz *support count* od 470).

Pravila generirana u Splunku jednaka su pravilima generiranima u Weki. Postoje minimalne razlike u pouzdanostima no Weka zaokružuje vrijednosti na 2 decimale za razliku od Splunka koji ne zaoukružuje vrijednosti te se zapravo radi o jednakim iznosima pouzdanosti.

Instalacija programske podrške

Korištenje razvijene aplikacije opisane u ovom radu zahtjeva instalaciju instance Splunka lokalno na korisnikovom računalu. Instalacijski paket Splunka moguće je preuzeti na službenim stranicama, a dostupan je za većinu operativnih sustava (Windows 8, 8.1, Linux, OSX, FreeBSD…).

Nakon uspješne instalacije Splunka, potrebno je direktorij imena 'association_rules/' koji se nalazi na optičkom mediju priloženom uz rad iskopirati u direktorij koji se nalazi na sljedećoj putanji '<putanja direktorija na vašem računalu u kojem se nalazi instalirana instanca Splunka>/Splunk/etc/apps/'.

Direktorij 'association_rules/' je Splunkova aplikacija koju je moguće koristiti nakon što se njen sadržaj nalazi na potrebnoj lokaciji koja je prethodno navedena.

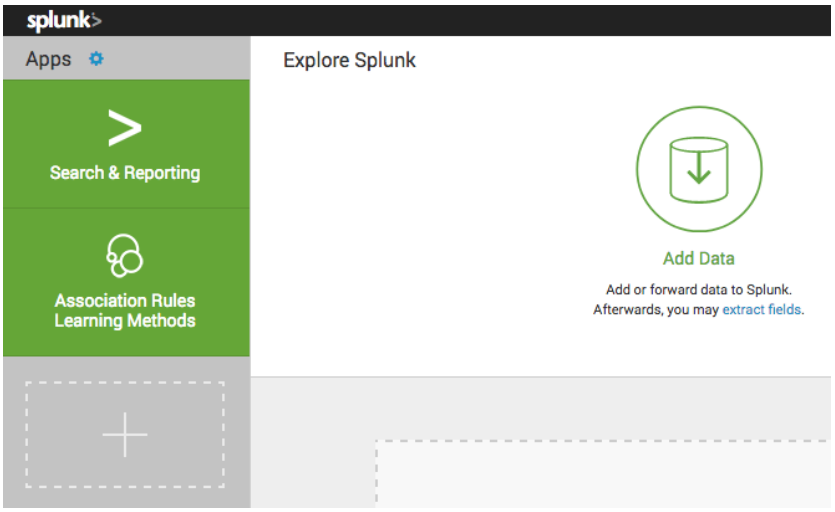
Ovime je postavljeno razvojno okruženje potrebno za rad aplikacije.

Upute za korištenje programske podrške

Nakon uspješne instalacije Splunka i aplikacije potrebno se pozicionirati u direktorij na putanji '<putanja direktorija na vašem računalu u kojem se nalazi instalirana instanca Splunka>/Splunk/bin/' te pokrenuti bash skriptu imena 'splunk' uz dodatno napisani argument 'start' čime pokrećemo Splunk. U konzoli će potom biti napisana adresa na kojoj je dostupno korisničko sučelje Splunka, većinom se radi o adresi http://127.0.0.1:8000 (8000 je broj vrata (engl. *port*)).

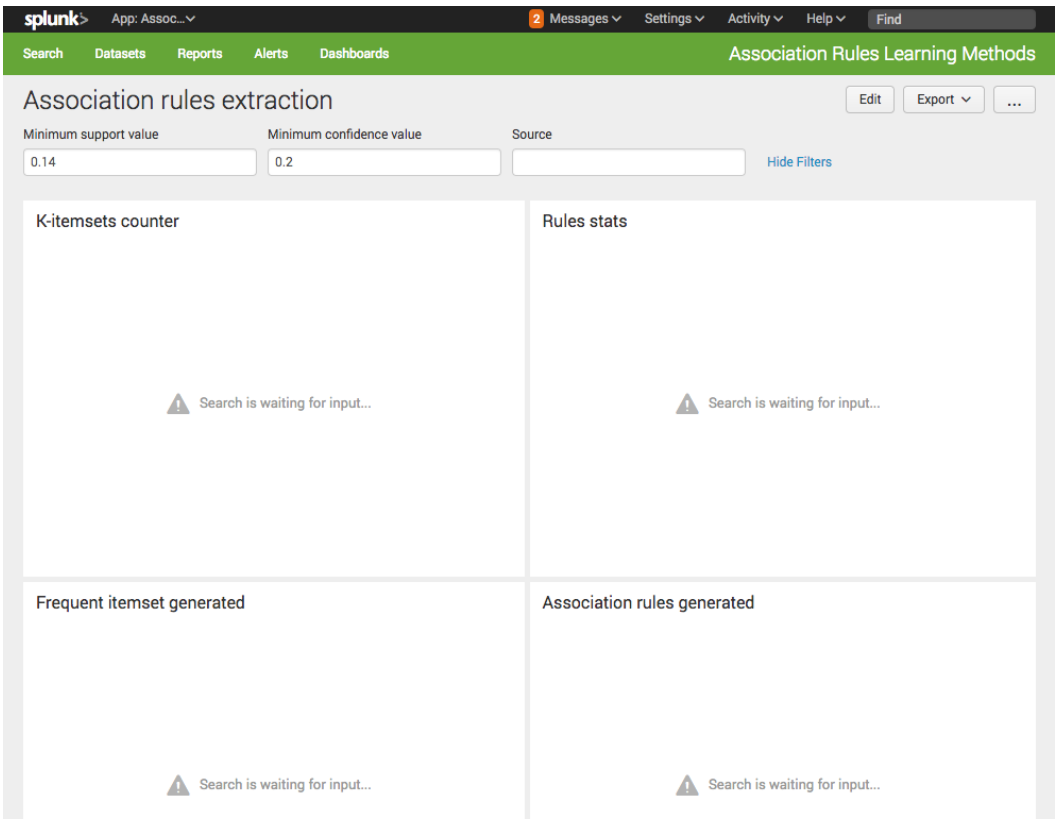
Otvaranjem stranice na prethodnoj adresi pokrećemo web sučelje Splunka. Prilikom prvog otvaranja potrebno se prijaviti sa korisničkim imenom i lozinkom koji su dani na zaslonu te potom unesti željene vrijednosti korisničkog imena i lozinke koji će se u budućnosti koristiti za prijavu.

Nakon uspješne prijave otvara se početni zaslon Splunka na kojem sa lijeve strane u padajućem izborniku možemo odabrati našu aplikaciju koja se zove 'Association Rules Learning Methods' ili ako želimo učitati novi set podataka odabiremo link pod sličicom 'Add Data'. Prikaz početnog zaslona dan je u nastavku.



Slika 29: Početni zaslon Splunka

Nakon uspješno učitanog skupa podataka (potrebno je slijedno pratiti korake koji su logični i dobro formulirani) potrebno je zapamtiti *source* vrijednost koja je vidiljiva u poslijednjem koraku učitavanja podataka jer s njom u aplikaciji pozivamo željeni učitani set podataka. Nakon unosa *sourcea* odabiru se proizvoljne vrijednosti minimalne signifikantnosti i pouzdanosti poslije čega Splunk automatski pokreće analizu.



Slika 30: Početni zaslon aplikacije unutar Splunka

Zaključak

Zadatak ovog rada bio je upoznati se s postupcima za učenje asocijativnih pravila te napraviti vlastitu implementaciju istih uz usporedbu konačnih rješenja sa već postojećim implementacijama.

Nakon provedene usporedbe mogu reći kako na nekim mjestima nisam očekivao nastale razlike (par pravila manje generiranih kroz Splunk nego Weku ili manje razlike u broju *k*-itemsetova na određenim setovima podataka) koje su se pojavile, no ako uzmemo u obzir kako su algoritmi implementirani u Weki optimizirani do najmanjeg detalja, nastale razlike zapravo i nisu toliko neočekivane.

Prilikom prvog susreta s asocijativnim pravilima i algoritmom Apriori pomislio sam kako će implementacija istoga biti puno lakši posao nego što se naposlijetku ispostavilo. Ideja koja stoji iza algoritma i postupak implementacije se na prvi pogled ne čine kompliciranima, no kada nakon svog provedenog rada nad algoritmom i uvjerenja kako implementacija nije mogla biti točnije napravljena ipak dođe do razlika rješenja, u usporedbi sa implementacijom za čija rješenja možemo biti sigurni kako su točna, vidljivo je koliko zapravo postupak implementacije takvog algoritma zapravo nije trivijalan.

Ako malo razmislimo o pravilima koja generiraju postupci za učenje asocijativnih pravila, uzmimo za primjer pravilo `windy=false → play=YES`. Dano pravilo ima poprilično istinito značenje do kojeg ponekad dolazimo isključivo vlastitim zaključivanjem, a u ovom slučaju do isto zaključka je došao i jedan programski algoritam.

Ovisno o skupu ulaznih podataka asocijativna pravila mogu proizvesti pravila široke životne primjene, primjerice na temelju povijesnih podataka o stanju vremena generiraju pravilo kojim sa velikom pouzdanošću možemo predvidjeti buduću pojavu velike ljetne oluje te na vrijeme upozoriti stanovnike.

Marin Smoljanić

Literatura

[1] WIKIPEDIA, *Apriori Algorithm.* Dostupno na: https://en.wikipedia.org/wiki/Apriori_algorithm

[2] DMS, *Association Rules*. Dostupno na: http://dms.irb.hr/tutorial/hr_tut_assoc_rules.php

[3] DAVID C., *Splunk Documentation: Exploring Splunk*

[4] PAN-NING T., MICHAEL S., VIPIN K., *Introduction to Data Mining*, University of Minnesota & Michigan State University (2006.)

[5] SPLUNK, Quick reference guide. Dostupno na: http://docs.splunk.com/

[6] INFIGO, Splunk-opis proizvoda. Dostupno na: http://www.infigo.hr/splunk-s85

Sažetak

Postupci za učenje asocijativnih pravila

Asocijativna pravila pogodna su za pronalaženje relacijskih odnosa između više varijabli u velikim skupovima podataka. Odnosi između varijabli, takozvana asocijativna pravila, izvode se u obliku implikacija AKO-ONDA.

Najčešće korišteni algoritmi generiranja asocijativnih pravila su Apriori, FP-Growth, Tertius, Eclat i OPUS.

U okviru ovog rada opisani su principi rada algoritama Apriori i FP-Growth te je opisan postupak implementacija algoritma Apriori unutar sustava za upravljanje podatcima imena Splunk.

U konačnici vlastita implementacija algoritma Apriori uspoređena je na više skupova podataka sa postojećom implementacijom istog algoritma u aplikaciji imena Weka.

Ključne riječi:

Asocijativna pravila, Apriori, FP-Growth, signifikantonst, pouzdanost, Weka, Splunk, Python

Summary

Association rules learning methods

The association rules are suitable for finding relationships between multiple variables in large sets of data. Relationships between variables, so-called association rules, are derived in the form of IF-THEN implications.
The most commonly used algorithms for generating association rules are Apriori, FP-Growth, Tertius, Eclat and OPUS.

This thesis describes the principles of Apriori and FP-Growth algorithms and also describes the procedure of Apriori algorithm implementation within Splunk data management system.

Results of Splunk implementation of the Apriori algorithm are compared to multiple data sets with the existing implementation of the same algorithm in the Weka application.

Keywords:

Association rules, Apriori, FP-Growth, support, support count, itemset, confidence, Weka, Splunk, Python