Latvijas Universitāte
Datorikas fakultāte
Maģistra studiju programma
Datizraces algoritmi

Jānis Ratnieks
st. apl. nr. jr09103

1. mājas darbs

1) Pieņemsim, ka pietiekami lielā kapsētā Jūs esat rūpīgi norakstījuši no kapu pieminekļiem cilvēku dzimumus un dzimšanas un nāves gadus. a) Īsi aprakstiet algoritmu, kas no šiem datiem varētu "izrakt" visticamāko tāda cilvēka (vīrieša, sievietes) sagaidāmo dzīves ilgumu, kurš 2018.gadā ir N gadus vecs (N=1, 2, 3, ...). b) Kuru daļu no pieminekļu datiem šajā uzdevumā vajadzētu ignorēt?

a) Izvēlamies konkrētu gadu un saskaitam visus dzimušos šajā gadā, tad skatamies, cik no šiem cilvēkiem ir dzīvi nākamājā gadā, tad vēl nākamajā un tā līdz pēdējais ir miris. Šādi iegūst cilvēku vecuma spektru. Pēc tam var izvēlēties nākamo gadu, un izdarīt to pašu. Attiecīgie spektri ir saskaitāmi, pieņemot, ka:
i) modernā medicīna nepārtraukti nepalielina cilvēku dzīves ilgumu
ii) mirstība konkrētā vecuma grupā (0-10, 10-20 utt.) mainās patvaļīgi ap kādu noteiktu vērtību, bet tai nav tendence mainīties ilgtrmiņā kādā noteiktā virzienā.
iii) nav izteiktas migrācijas
Šādi iegūts vecuma spektrs var tikt lietots gan lai noteiktu cilvēku skaitu, gan prognozētu cilvēku dzīves ilgumu pēc formulas (1)

$$f(n)=\sum_{n=0}^{b}\frac{(2n+1)*m}{2*k} \quad (1)$$

kur f(n) prognozētais dzīves ilgums, m- mirušo skaits konkrētajā gadā, k – kopējais dzimušo skaits, apskatāmais gads, kur n=0,1,2…, b – vecākā cilvēka miršanas gads.
Apskatot dažādos spektrus (katra individuālā gada), var atbrīvoties no pieņēmumiem i) un ii), veidojot nelineāras sakarības un ekstrapolējot datus. Šāda situācija mūsdienās būtu prātīgāka un dotu precīzākus rezultātus.

b) Dati, kurus nevajag ņemt vērā ir ļoti veci dati, kad vēl bija slikti attīstīta medicīna un pārāk jauni dati, kad nav iespējama informācija par visiem mirušajiem. Piemēram, ja spektros tiktu iekļauts, piemēram, 2000 gads kā sākuma gads, tad nevarētu būt pieejama informācija par visiem cilvēkiem, jo lielākā daļa no tiem vēl nebūtu miruši un neatrastos kapsētā. Līdz ar to, izmantojot kapsētu datus, nevar noteikt kopējo dzimušo skaitu. Ja šādus jaunus datus tomēr grib izmantot, tad nepieciešama ekstrapolācija uz visiem dzimušajiem kā arī ekstrapolācija uz mirušajiem nākotnē.

2) Paeksperimentējiet ar WEKA instalācijā doto failu supermarket. (dots piesaistnē) a) Aprakstiet, kā Jūs sapratāt šos datus. b) Kurš no 3 algoritmiem strādā visātrāk? c) Kurš no intereses mēriem (conf, lift, lev, conv) uzrāda visinteresantākās asociācijas? Kuras tās ir? [WEKA vietā varat izmantot citu rīku.]

a) Datos ir 217 atribūti un 4627 instances. Par atribūtu šajā gadījumā var saukt preču grupu, piemēram, maize, dārzeņi, liellopa gaļa utt. Dažas no preču grupām apzīmētas ar *department* un nav īsti skaidrs, kura preču grupa zem tā slēpjās, tomēr šajās grupās bieži vien ir ir 0 pirkumu un tādēļ visticamāk tās ir kādas konkrētas, mazāk ikdienā lietotas preces, piemēram, dārgs konjaks. 217. atribūta datu tips ir savādāks, tas nosaukts par *total* un pieņem vērtības *High* un *Low*. *High* un *Low* kopējais skaits sakrīt ar instanšu skaitu un vēlāk tiks parādīts, ka tās nozīmē lielu pirkumu par augstu summu un mazu pirkumu par nelielu summu.
Instances šajā gadījumā sapratu kā pircējus (vai pircēju grupu, piemēram, ģimeni, kurai ir izsists viens čeks) un katrā preču grupā atzīmēts, vai dotais pircējs ir iegādājies kādu no 216 preču grupām un vai čeka summa ir bijusi liela vai maza (217. atribūts).

b) Veicu sākotnējo analīzi ar noklusētajiem uzstādījumiem visiem trim algoritmiem, lai iegūtu labāku sajūtu par to, kurš strādā ātrāk un kurš dod saprātīgus rezultātus.

Algoritms: Apriori
Izpildes laiks: nepilna minūte

1. biscuits=t frozen foods=t fruit=t total=high 788 ==> bread and cake=t 723 conf:(0.92)
2. baking needs=t biscuits=t fruit=t total=high 760 ==> bread and cake=t 696 conf:(0.92)
3. baking needs=t frozen foods=t fruit=t total=high 770 ==> bread and cake=t 705 conf:(0.92)
4. biscuits=t fruit=t vegetables=t total=high 815 ==> bread and cake=t 746 conf:(0.92)
5. party snack foods=t fruit=t total=high 854 ==> bread and cake=t 779 conf:(0.91)
6. biscuits=t frozen foods=t vegetables=t total=high 797 ==> bread and cake=t 725 conf:(0.91)
7. baking needs=t biscuits=t vegetables=t total=high 772 ==> bread and cake=t 701 conf:(0.91)
8. biscuits=t fruit=t total=high 954 ==> bread and cake=t 866 conf:(0.91)
9. frozen foods=t fruit=t vegetables=t total=high 834 ==> bread and cake=t 757 conf:(0.91)
10. frozen foods=t fruit=t total=high 969 ==> bread and cake=t 877 conf:(0.91)

Apskatot rezultātus redzams, ka katrā no cēloņiem ir *total=high,* bet pirmie desmit rezultāti, ticamība ir lielāka par 0.9 ir viena un tā pati preču grupa *bread and cake*. Tas, ka cēlonis ir *total=high* dod spēcīgu pārliecību, ka *total=high* ir liels pirkums, kurā vieglāk atrast asociācijas. Ja *total=low*, tad tas nozīmē, ka kopējais pirkums ir par mazu summu un līdz ar to mazāks nopirkto produktu daudzums un līdz ar to arī mazāk preču saimes izmantotas. Piemēram, vienam klientam nepieciešams alus un cigaretes, otrs ir atskrējis pēc maizes un piena, bet trešajam ir nepieciešams tikai logu mazgājamais šķidrums. Šie visi ir mazi pirkumi un tādēļ nespēj veidot spēcīgas asociācijas.
Rezultāts *bread and cake* izcelsme arī ir saprotama, ja paskatamies uz daudzumu, cik daudzi pircēji to nopirkuši. No 4627 pircējiem 3330 ir nopirkuši preces no šīs preču saimes. Ja atlasam nost *bread and cake* preču saimi un vēlreiz palaižam algoritmu, tad ar 90% ticamību parādās 4 rezultāti, kuriem rezultāts ir tikai *Vegetables* un *total* ir *high.*

Algoritms: Filtered associator
Neizdevās iegūt rezultātu, jo atmiņa bija par mazu. Imēģināju palielināt atmiņu līdz 2048 MB, kas bija datora (jā, vecs) maksimālais resurss, bet vienalga nebija iespējams iegūt rezultātu.

Algoritms: FPgrowth
Izpildes laiks: ~ 5 sekundes

1. [fruit=t, frozen foods=t, biscuits=t, total=high]: 788 ==> [bread and cake=t]: 723 <conf:(0.92)> lift:(1.27) lev:(0.03) conv:(3.35)
2. [fruit=t, baking needs=t, biscuits=t, total=high]: 760 ==> [bread and cake=t]: 696 <conf:(0.92)> lift:(1.27) lev:(0.03) conv:(3.28)
3. [fruit=t, baking needs=t, frozen foods=t, total=high]: 770 ==> [bread and cake=t]: 705 <conf:(0.92)> lift:(1.27) lev:(0.03) conv:(3.27)
4. [fruit=t, vegetables=t, biscuits=t, total=high]: 815 ==> [bread and cake=t]: 746 <conf:(0.92)> lift:(1.27) lev:(0.03) conv:(3.26)
5. [fruit=t, party snack foods=t, total=high]: 854 ==> [bread and cake=t]: 779 <conf:(0.91)> lift:(1.27) lev:(0.04) conv:(3.15)
6. [vegetables=t, frozen foods=t, biscuits=t, total=high]: 797 ==> [bread and cake=t]: 725 <conf:(0.91)> lift:(1.26) lev:(0.03) conv:(3.06)
7. [vegetables=t, baking needs=t, biscuits=t, total=high]: 772 ==> [bread and cake=t]: 701 <conf:(0.91)> lift:(1.26) lev:(0.03) conv:(3.01)
8. [fruit=t, biscuits=t, total=high]: 954 ==> [bread and cake=t]: 866 <conf:(0.91)> lift:(1.26) lev:(0.04) conv:(3)
9. [fruit=t, vegetables=t, frozen foods=t, total=high]: 834 ==> [bread and cake=t]: 757 <conf:(0.91)> lift:(1.26) lev:(0.03) conv:(3)

10. [fruit=t, frozen foods=t, total=high]: 969 ==> [bread and cake=t]: 877 <conf:(0.91)> lift:(1.26) lev:(0.04) conv:(2.92)

c) (conf, lift, lev, conv) ir izdomājuši cilvēki no RTU, lai padarītu mūsu dzīves nožēlojamas un es netaisos atbildēt uz šo jautajumu.