Розділ 4.

АСОЦІАТИВНІ ПРАВИЛА ТА ДЕРЕВА РІШЕНЬ

4.1. Основні поняття теорії асоціативних правил.

Пошук асоціативних правил (Association Rules) - ключова тема в інтелектуальному аналізі даних. Асоціація має місце в тому випадку, якщо декілька подій зв'язані одна з одною. Наприклад, дослідження, проведене в супермаркеті, може показати, що 65% відвідувачів, які купили кукурудзяні чіпси беруть також і «кока-колу», а за наявності знижки за такий комплект купують його в 85% випадків. Маючи в своєму розпорядженні відомості про подібну асоціацію, менеджерам легко оцінити, наскільки дієва знижка, що надається.

Пошук виявляє приховані зв'язки в, на перший погляд, ніяк незв'язаних даних. Ці зв'язки - правила. Ті, які перевищують певний поріг, вважаються цікавими. Такі правила дають можливість виконувати дії ґрунтуючись на певних шаблонах. Вони так само допомагають в прийнятті і поясненні рішень. Як і більшість методів data mining, даний метод дозволяє перетворити потенційно величезну кількість інформації в невеликий і зрозумілий набір статистичних показників [9, , 37, 46, 55, 68].

Одним з найчастіше цитованих прикладів пошуку асоціативних правил служить проблема пошуку стійких зв'язків в корзині покупця (Market Basket Problem). Проблема пошуку стійких зв'язків в корзині покупця полягає в тому, аби визначити які товари отримуються покупцями разом, так, щоб фахівці з маркетингу могли відповідним чином розмістити ці товари в магазині для підвищення об'єму продажів, і так само прийняти інші рішення, сприяючі продажам. Деякі правила, що виявляються, можуть бути тривіальними, наприклад, «покупці, які купують хліб, так само купують і масло». Інші - цікаві і екстраординарні, наприклад «покупці, які купують дитячі товари, так само

купують і пиво». Наведемо один відомий приклад. Менеджери одного з супермаркетів найбільшої міжнародної мережі Wal - Mart, завдяки технології data mining, виявили, що в п'ятницю увечері пиво чомусь особливо добре продається разом з дитячими товарами. Спочатку вони були украй здивовані: здавалося б, який може бути зв'язок між такими різними товарами? Незабаром, проте, пояснення знайшлося: багато чоловіків, повертаючись вечорами з роботи, на прохання дружин, купували своїм дітям приналежності; а в п'ятницю при цьому вони помічали, що за важкий трудовий тиждень заслужили на свою винагороду – і додавали в корзину пиво. Менеджери Wal - Mart уміло скористалися такою знахідкою, поставивши поряд з дитячими товарами одні з найдорожчих марок пива і добилися значного зростання його продажів. Інший приклад - річна економія 700 тис. дол. за рахунок впровадження data mining в мережі універсамів у Великобританії. Саме здатність виявляти цікаві правила робить пошук асоціативних правил важливим процесом, що сприяє пошуку знань (рис. 4.1).

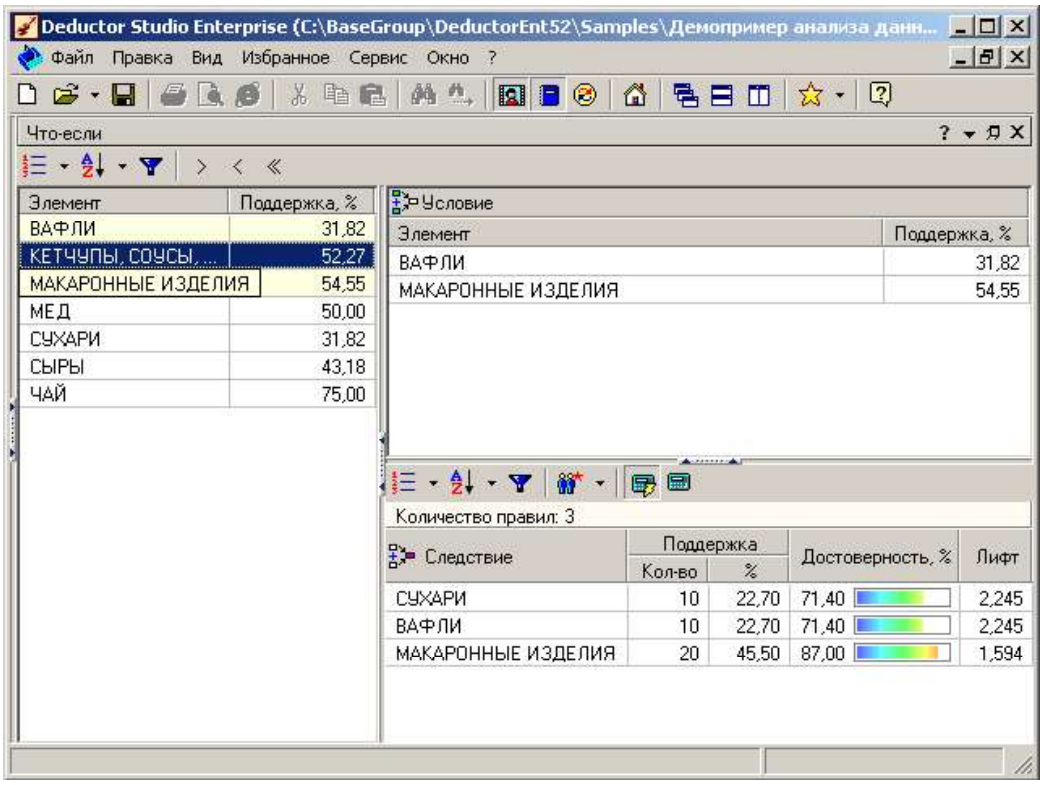


Рис. 4.1. Приклад застосування асоціативних правил.

Перший алгоритм пошуку асоціативних правил, що називався AIS, був розроблений в 1993 році співробітниками дослідницького центру IBM Almaden. З цієї піонерської роботи зріс інтерес до асоціативних правил; на середину 90-х років минулого століття припав пік дослідницьких робіт в цій області, і з тих пір щороку з'являлося по декілька нових алгоритмів. Вперше це завдання було запропоноване як пошук асоціативних правил для знаходження типових шаблонів покупок, що здійснюються в супермаркетах, тому інколи його ще називають аналізом ринкової корзини (market basket analysis). Проте сфера застосування цих алгоритмів не обмежується лише однією торгівлею. Їх також успішно застосовують і в інших областях: медицині, для аналізу відвідин веб-сторінок (Web Mining), для аналізу тексту (Text Mining), для аналізу даних по перепису населення, в аналізі і прогнозуванні збоїв телекомунікаційного устаткування і так далі. Завданням пошуку асоціативних правил не є виявлення всіх правил, оскільки частина з них відомі аналітикам, інші можуть не представляти статистичної цінності. Тому при пошуку вводяться пороги підтримки і достовірності асоціативних правил [2, 25, , 47, 56].

Нехай є база даних, що складається з купівельних транзакцій. Кожна транзакція - це набір товарів, куплених покупцем за один візит. Таку транзакцію ще називають ринковою корзиною.

Означення 1. Хай $I=(i_1,i_2,...,i_n)$ - множина (набір) товарів, званих елементами. Хай D - множина транзакцій, де кожна транзакція T - це набір елементів з I, $T\subseteq I$. Кожна транзакція є бінарним вектором, де $t[k]=1$, якщо i_k елемент присутній в транзакції, інакше $t[k]=0$. Вважатимемо, що транзакція T містить X, деякий набір елементів з I, якщо $X\subset T$. Асоціативним правилом називається імплікація $X\Rightarrow Y$, де $X\subset I$, $Y\subset I$ і $X\cap Y=0$. Правило $X\Rightarrow Y$ має підтримку s (support), якщо $s\%$ транзакцій з D, містять $X\cup Y$, $\sup p(X\Rightarrow Y)=\sup p(X\cup Y)$. Достовірність правила показує яка ймовірність того, що з X виходить Y. Правило $X\Rightarrow Y$ справедливо з достовірністю c

(confidence), якщо $c\%$ транзакцій з D, що містять X, також містять Y, $conf\,(X \Rightarrow Y) = \sup p(X \cup Y) / \sup p(X)$.

Розглянемо визначення на конкретному прикладі: «75% транзакцій, що містять хліб, також містять і молоко. 3% від загального числа всіх транзакцій містять обоє товари». 75% - це достовірність (confidence) правила, 3% це підтримка (support), або «Хліб»-«Молоко» з вірогідністю 75%. Іншими словами, метою аналізу є встановлення наступних залежностей: якщо в транзакції зустрівся деякий набір елементів X, то на підставі цього можна зробити висновок про те, що інший набір елементів Y також повинен з'явитися в цій транзакції. Встановлення таких залежностей дає нам можливість знаходити прості і інтуїтивно зрозумілі правила. Алгоритми пошуку асоціативних правил призначені для знаходження всіх правил X, Y, причому підтримка і достовірність цих правил мають бути вище за деякі наперед задані пороги, звані відповідно мінімальною підтримкою (minsupport) і мінімальною достовірністю (minconfidence). Якщо значення підтримки правила дуже велике, то в результаті роботи алгоритму будуть знайдені правила очевидні і добре відомі. Дуже низьке значення підтримки приведе до знаходження дуже великої кількості правил, які, можливо, будуть в більшій частині необґрунтованими, але і не відомими і не очевидними для аналітика. Таким чином, необхідно визначити такий інтервал, «золоту середину», який з одного боку забезпечить знаходження неочевидних правил, а з іншої - їх обґрунтованість. Якщо рівень достовірності дуже малий, то цінність правила викликає серйозні сумніви. Наприклад, правило з достовірністю в 3% лише умовно можна назвати правилом.

Пошук асоціативних правил зовсім не тривіальна задача, як може здатися на перший погляд. Одна з проблем - алгоритмічна складність при знаходженні часто зустрічаючих наборів елементів, оскільки із зростанням числа елементів в I експоненціально зростає число потенційних наборів елементів.

Існують різні типи асоціативних правил. У простій формі асоціативні правила повідомляють лише про наявність або відсутність асоціації. Логічна природа таких правил озвучена в їх назві - *булеві асоціативні правила* (Boolean Association Rule). На прикладі корзини споживача це відбувається так, «споживачі, які купують зняте молоко так само купують масло з низьким рівнем жиру», - типове булеве асоціативне правило.

Правила, які збирають декілька асоціативних правил разом, називаються *мультирівневі* або *узагальнені асоціативні правила* (Multilevel or Generalized Association Rules). При побудові таких правил елементи зазвичай групуються згідно ієрархії і пошук ведеться на найвищому концептуальному рівні. Наприклад, «споживачі, які купують молоко, так само купують хліб". В даному прикладі, «молоко і хліб» містять ієрархію різних типів і брендів, проте пошук на нижньому рівні не дозволить знайти цікаві правила.

Розглянемо цей процес детальніше. При пошуку асоціативних правил передбачалось, що всі аналізовані елементи однорідні. Повертаючись до аналізу ринкової корзини, це товари, що мають абсолютно однакові атрибути, за винятком назви. Проте не складе великих труднощів доповнити транзакцію інформацією про те, до якої товарної групи входить товар і побудувати ієрархію товарів. Наведемо приклад такого угрупування (таксономії) у вигляді ієрархічної моделі (рис. 4.2).

Нехай нам дана база транзакцій D і відомо в які групи (таксони) входять елементи. Тоді можна витягувати з даних правила, що пов'язують групи з групами, окремі елементи з групами і так далі. Наприклад, якщо покупець купив товар з групи «Безалкогольні напої», то він купить і товар з групи «Молочні продукти». Це і є узагальнені асоціативні правила.

Визначення 2. Узагальненим асоціативним правилом називається імплікація $X \Rightarrow Y$, де $X \subset I$, $Y \subset I$ і $X \cap Y = 0$ і де жоден з елементів, що входять в набір Y, не є предком жодного елементу, що входить в X. Підтримка і достовірність підраховуються так само, як і в разі асоціативних правил

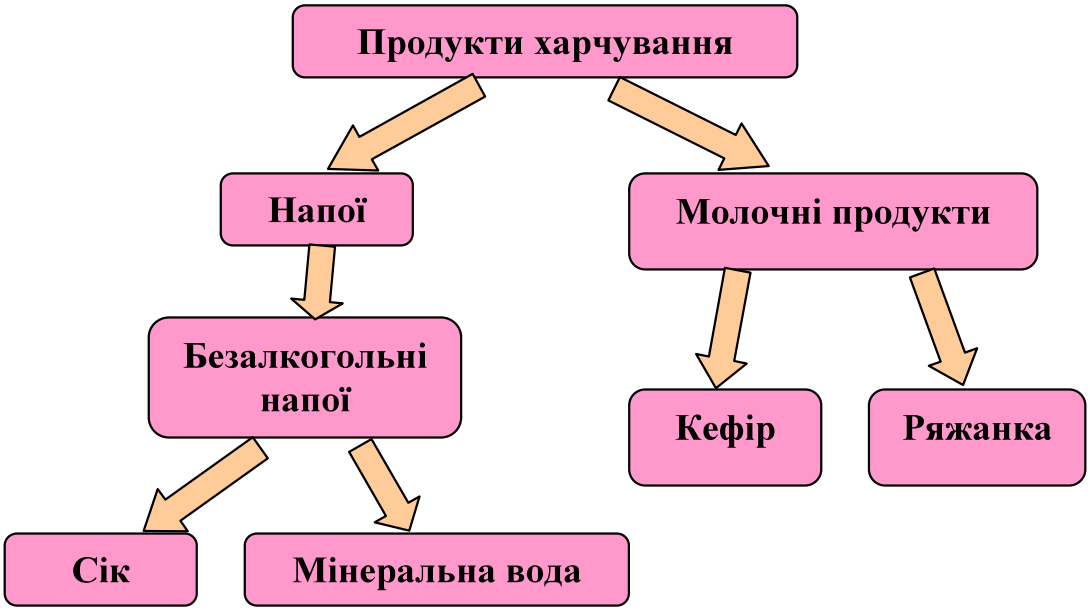


Рис. 4.2. Приклад ієрархічної моделі товарів.

Введення додаткової інформації про угрупування елементів у вигляді ієрархії дасть наступні переваги:

1. допомагає встановити асоціативні правила не лише між окремими елементами, але і між різними рівнями ієрархії (групами);

2. окремі елементи можуть мати недостатню підтримку, але в цілому група може задовольняти порогу minsupport.

Для знаходження таких правил можна використовувати будь-який з стандартних алгоритмів. Для цього кожну транзакцію потрібно доповнити всіма предками кожного елементу, що входить в транзакцію. Проте, застосування «в лоб» цих алгоритмів неминуче приведе до наступних проблем:

1. елементи на верхніх рівнях ієрархії прагнуть до значно великих значень підтримки в порівнянні з елементами на нижніх рівнях;

2. з додаванням в транзакції груп збільшилася кількість атрибутів і відповідно розмірність вхідного простору. Це ускладнює задачу, а також веде до генерації більшої кількості правил;

3. поява надлишкових правил, що перечать визначенню узагальненого асоціативного правила, наприклад, «Сік» - «Прохолодні напої». Вочевидь, що

практична цінність такого «відкриття» нульова при 100% достовірності. Отже, потрібні спеціальні оператори, що видаляють подібні надлишкові правила. Для знаходження узагальнених асоціативних правил бажане використання спеціалізованого алгоритму, який усуває вищеописані проблеми і до того ж працює в 2-5 разів швидше, ніж стандартний.

Складнішим типом правил є *кількісні асоціативні правила* (Quantitative Association Rules). Цей тип правил шукається із застосуванням кількісних (наприклад, ціна) або категоріальних (наприклад, стать) атрибутів. Наприклад, «покупці, чий вік знаходиться між 30 і 35 роками з доходом більше 75000 в рік купують машини вартістю більше 20000». Однією з ключових проблем пошуку таких правил є дискретизація числових атрибутів, яка полягає в заміні безперервного атрибуту його дискретним аналогом. Методика дискретизації впливає на цікавість виявлених правил, втрату правил унаслідок не задоволення порогам підтримки або достовірності, час роботи алгоритму. Дискретизацію числових атрибутів може проводити експерт прикладної області задаючи розбиття діапазону значень числового атрибуту на інтервали, ґрунтуючись на своїх знаннях з прикладної області.

Вищеперелічені типи правил не зачіпають той факт, що транзакції, за своєю природою, залежать від часу. Наприклад, пошук до того, як продукт був виставлений на продаж або після того, як він зник з ринку, несприятливо вплине на порогове значення підтримки (support). З врахуванням цього, введена концепція атрибутного часу життя в алгоритмах пошуку *часових асоціативних правил* (Temporal Association Rules).

Окрім описаних вище асоціативних правил існують *непрямі асоціативні правила, асоціативні правила із запереченням* та інші.

Не дивлячись на різні типи правил, алгоритм для пошуку асоціативних правил може бути в загальному вигляді розділений на два етапи:

1. пошук найбільш часто зустрічающихся наборів елементів (large (frequent) itemsets). Набір, що часто зустрічається, - це набір, в якого підтримка перевищує мінімальне значення;

2. генерація правил на основі часто зустрічающихся наборів.

З моменту появи, алгоритм Apriori найчастіше застосовується на першому кроці. Другий крок в основному характеризується достовірністю і цікавістю. Є дослідження, присвячені пошуку інших доріг генерації правил і альтернативним вимірам цікавості. Крім того, існує ряд досліджень присвячених генеруванню інших типів правил.

Алгоритм AIS. Першим алгоритмом пошуку асоціативних правил був алгоритм AIS, запропонований співробітниками дослідницького центру IBM Almaden Agrawal, Imielinski та Swami в 1993 році. З цієї роботи і почався інтерес до асоціативних правил. У алгоритмі AIS кандидати множини наборів генеруються і підраховуються «на льоту», в процесі сканування бази даних. Кожна транзакція перевірялася на наявність великих наборів, виявлених при попередньому проході. Відповідно, нові набори формувалися шляхом розширення наявних наборів.

Алгоритм SETM. Створення цього алгоритму було мотивоване бажанням використовувати мову SQL для обчислення часто зустрічающихся наборів товарів. Як і алгоритм AIS, SETM також формує кандидатів «на льоту», ґрунтуючись на перетвореннях бази даних. Аби використовувати стандартну операцію об'єднання мови SQL для формування кандидата, SETM відділяє формування кандидата від їх підрахунку.

Незручність алгоритмів AIS і SETM - зайве генерування і підрахунок дуже багатьох кандидатів, які в результаті не виявляються такими, що часто зустрічаються. Для поліпшення їх роботи був запропонований алгоритм Apriori.

Алгоритм Apriori. Робота даного алгоритму складається з декількох етапів, кожен з яких складається з наступних кроків:

- формування кандидатів;
- підрахунок кандидатів.

Формування кандидатів (candidate generation) - етап, на якому алгоритм, скануючи базу даних, створює множину i-елементних кандидатів (i - номер етапу). На цьому етапі підтримка кандидатів не розраховується. *Підрахунок*

кандидатів (candidate counting) - етап, на якому обчислюється підтримка кожного i-елементного кандидата. Тут же здійснюється відсікання кандидатів, підтримка яких менше мінімуму, встановленого користувачем (minsup). I-елементні набори, що залишилися, називаємо такими, що часто зустрічаються.

Для того, щоб було можливо застосувати алгоритм, необхідно провести передобробку даних: по-перше, привести всі дані до бінарного вигляду; по-друге, змінити структуру даних. Звичайний вигляд бази даних транзакцій представлен на рис. 4.3, а нормалізованої – на рис. 4.4.

Номер транзакції	**Найменування елементу**	**Кількість**
1001	A	2
1001	D	3
1001	E	1
1002	A	2
1002	F	1
1003	B	2
1003	A	2
1003	C	2
…	…	…

Рис. 4.3. Фрагмент основної бази даних.

TID	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**K**	**...**
1001	1	0	0	1	1	0	0	0	0	0	…
1002	1	0	0	0	0	1	0	0	0	0	…
1003	1	1	1	0	0	0	0	0	1	0	…

Рис. 4.4. Фрагмент нормалізованої бази даних.

Кількість стовпців в таблиці дорівнює кількості елементів, присутніх в множині транзакцій D. Кожен запис відповідає транзакції, де у відповідному стовпці стоїть 1, якщо елемент присутній в транзакції, і 0 інакше. Відмітимо, що вигляд таблиці може бути відмінним від приведеного на рис. 4.3. Головне, аби дані

були перетворені до нормалізованого вигляду, інакше алгоритм не може бути застосований. Більш того, як видно з таблиці, всі елементи впорядковані в алфавітному порядку (якщо це числа, вони мають бути впорядковані в числовому порядку).

Як було відмічено раніше, такі алгоритми працюють в два етапи. На першому кроці необхідно знайти набори елементів, що часто зустрічаються, а потім, на другому, витягувати з них правила. Кількість елементів в наборі називатимемо розміром набору, а набір, що складається з k елементів, – k-елементним набором.

Виявлення часто зустрічающихся наборів елементів – операція, що вимагає багато обчислювальних ресурсів і, відповідно, часу. Примітивний підхід до рішення даної задачі – простий перебір всіх можливих наборів елементів. Це зажадає 2^I операцій, де I – кількість елементів. Apriori використовує одну з властивостей підтримки, яке свідчить: підтримка будь-якого набору елементів не може перевищувати мінімальної підтримки будь-якої з його підмножин. Наприклад, підтримка 3-елементного набору {Хліб, Масло, Молоко} буде завжди менша або дорівнює підтримці 2-елементних наборів {Хліб, Масло} {Хліб, Молоко} {Масло, Молоко}. Річ у тому, що будь-яка транзакція, що містить {Хліб, Масло, Молоко}, також повинна містити {Хліб, Масло}, {Хліб, Молоко}, {Масло, Молоко}, причому зворотне не вірно. Ця властивість носить назву *антимонотонності* і служить для зниження розмірності простору пошуку. Не май ми в наявності такої властивості, знаходження багатоелементних наборів було б практично нездійсненним завданням у зв'язку з експоненціальним зростанням обчислень.

Властивості антимонотонності можна дати і інше формулювання: із зростанням розміру набору елементів підтримка зменшується або залишається такою же. Зі всього вищесказаного виходить, що будь-який k-елементний набір буде таким, що часто зустрічається тоді і лише тоді, коли все його (k-1)-елементні підмножини будуть такими, що часто зустрічаються.

Всі можливі набори елементів з I можна представити у вигляді грат, що починаються з пустої множини, потім на 1 рівні 1-елементні набори, на 2 рівні – 2-елементні і так далі На k рівні представлені k-елементні набори, пов'язані зі всіма своїми (k-1)-елементними підмножинами.

Розглянемо рис 4.5, що ілюструє набір елементів I – {A, B, C, D}. Вважатимемо, що набір з елементів {A, B} має підтримку нижче заданого порогу і, відповідно, не є таким, що часто зустрічається. Тоді, згідно властивості антимонотонності, всі його супермножини також не є такими, що часто зустрічаються і відкидається. Вся ця гілка, починаючи з {A, B}, виділена фоном. Використання цієї евристики дозволяє істотно скоротити простір пошуку.

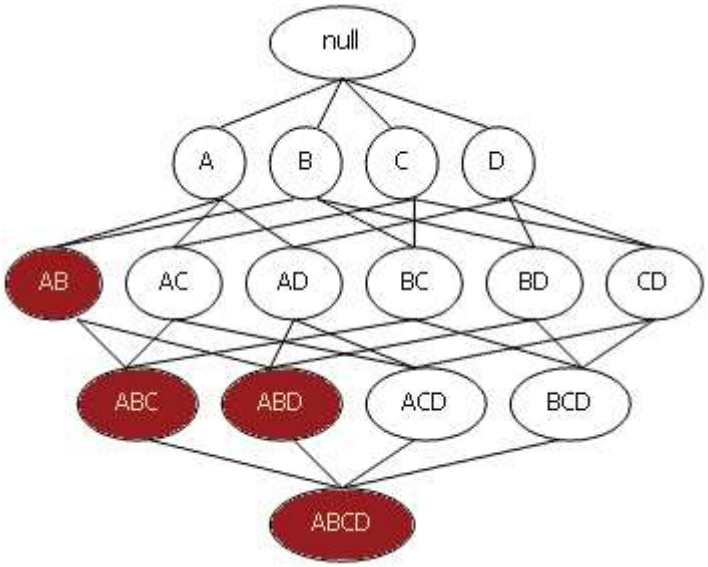


Рис. 4.5. Механізм скорочення простору пошуку.

На першому кроці алгоритму підраховуються 1-елементні набори, що часто зустрічаються. Для цього необхідно пройтися по всьому набору даних і підрахувати для них підтримку, тобто скільки разів зустрічається в базі. Наступні кроки складатимуться з двох частин: генерації потенційно часто зустрічающихся наборів елементів (їх називають кандидатами) і підрахунку підтримки для кандидатів. Описаний вище алгоритм можна записати у вигляді наступного псевдокоду:

1. F_1 = {1-елементні набори, що часто зустрічаються}
2. Для ($k = 2$; $F_{k-1} \lozenge \varnothing$; $k++$) {
3. C_k = Apriorigen(F_{k-1}) // генерація кандидатів
4. Для всіх транзакцій $t \in T$ {
5. C_t = subset(C_k, t) // видалення надлишкових правил
6. Для всіх кандидатів $c \in C_t$
7. c.count ++
8. }
9. $F_k = \{c \in C_k \mid c.count \geq \min sup\ port\}$ // відбір кандидатів
10. }
11. Результат $\cup F_k$

Опишемо функцію генерації кандидатів. Генерація кандидатів також складатиметься з двох кроків.

1. Об'єднання. Кожен кандидат C_k формуватиметься шляхом розширення часто зустрічающегося набору розміру $(k-1)$, шляхом додаванням елементу з іншого $(k-1)$-елементного набору. Приведемо алгоритм функції Apriorigen у вигляді невеликого SQL-подібного запиту.

   insert into C_k

   select p.item$_1$, p.item$_2$,…, p.item$_{k-1}$, q.item$_{k-1}$

   From F_{k-1} p, F_{k-1} q

   where p.item$_1$=q.item$_1$, p.item$_2$=q.item$_2$,…, p.item$_{k-1}$<q.item$_{k-1}$
2. Видалення надлишкових правил. На підставі властивості анти-монотонності, слід видалити всі набори $c \in C_k$ якщо хоч би одна з його $(k-1)$ підмножин не є такою, що часто зустрічається.

Після генерації кандидатів наступним завданням є підрахунок підтримки для кожного кандидата. Вочевидь, що кількість кандидатів може бути дуже великою і потрібний ефективний спосіб підрахунку. Найтривіальніший спосіб – порівняти кожну транзакцію з кожним кандидатом.

Але це далеко не краще рішення. Набагато швидше і ефективніше використовувати підхід, заснований на зберіганні кандидатів в хеш-дереві. Внутрішні вузли дерева містять хеш-таблиці з вказателями на нащадків, а листя – на кандидатів. Це дерево нами буде застосоване для швидкого підрахунку підтримки для кандидатів.

Хеш-дерево будується кожного разу, коли формуються кандидати. Первинне дерево складається лише з кореня, який є листом, і не містить жодних кандидатів-наборів. Кожного разу коли формується новий кандидат, він заноситься в корінь дерева і так до тих пір, поки кількість кандидатів в корені-листі не перевищить деякого порогу. Як тільки кількість кандидатів стає більше порогу, корінь перетвориться в хеш-таблицю, тобто стає внутрішнім вузлом, і для нього створюються нащадки-листя. І всі приклади розподіляються по вузлах-нащадках згідно хеш-значенням елементів, що входять в набір, і так далі Кожен новий кандидат хеширується на внутрішніх вузлах, поки він не досягне першого вузла-листа, де він і зберігатиметься, поки кількість наборів знову ж таки не перевищить порогу.

За допомогою хеш-дерева легко підрахувати підтримку для кожного кандидата. Для цього потрібно «пропустити» кожну транзакцію через дерево і збільшити лічильники для тих кандидатів, чиї елементи також містяться і в транзакції, тобто $C_k \cap T_i = C_k$. На кореневому рівні хеш-функція застосовується до кожного елементу з транзакції. Далі, на другому рівні, хеш-функція застосовується до других елементів і так далі На k-рівні хеширується k-елемент. І так до тих пір, поки не досягнемо листа. Якщо кандидат, що зберігається в листі, є підмножиною даної транзакції, тоді збільшуємо лічильник підтримки цього кандидата на одиницю. Після того, як кожна транзакція з вихідного набору даних «пропущена» через дерево, можна перевірити чи задовольняють значення підтримки кандидатів мінімальному порогу. Кандидати, для яких ця умова виконується, переносяться в розряд тих, що часто зустрічаються. Крім того, слід запам'ятати і підтримку набору, вона нам знадобиться при витяганні правил.

Після того, як знайдені всі набори елементів, що часто зустрічаються, можна приступити безпосередньо до генерації правил. Витягання правил – менш трудомістке завдання. По-перше, для підрахунку достовірності правила досить знати підтримку самого набору і множини, лежачої в умові правила. Наприклад, є часто зустрічающийся набір {A, B, C} і потрібно підрахувати достовірність для правила $AB \Rightarrow C$. Підтримка самого набору нам відома, але і його множина {A, B}, яка лежить в умові правила, також є такою, що часто зустрічається через властивість антимонотонності. Це означає, що його підтримка нам відома. Тоді ми легко зможемо підрахувати достовірність. Це позбавляє нас від небажаного перегляду бази транзакцій, що було б потрібно в тому разі якби ця підтримка була невідома.

Аби витягувати правило з набору F F, що часто зустрічається, слід знайти всі його непусті підмножини. І для кожної підмножини s ми зможемо сформулювати правило $s \Rightarrow (F - s)$, якщо достовірність правила $conf(s \Rightarrow (F - s) = \sup p(F) / \sup p(s)$ не менше порогу minconf. Відмітимо, що чисельник залишається постійним. Тоді достовірність має мінімальне значення, якщо знаменник має максимальне значення, а це відбувається у тому випадку, коли в умові правила є набір, що складається з одного елементу. Вся супермножина даної множини має меншу або рівнішу підтримку і, відповідно, більше значення достовірності. Ця властивість може бути використана при витяганні правил. Якщо ми почнемо витягувати правила, розглядаючи спочатку лише один елемент в умові правила, і це правило має необхідну підтримку, тоді всі правила, де в умові стоїть супермножина цього елементу, також мають значення достовірності вище заданого порогу. Наприклад, якщо правило $A \Rightarrow BCDE$ задовольняє мінімальному порогу достовірності minconf, тоді $AB \Rightarrow CDE$ також задовольняє. Для того, щоб витягувати всі правила використовується рекурсивна процедура. Важливе зауваження: будь-яке правило, складене з набору, що часто зустрічається, повинне містити всі елементи набору. Наприклад, якщо набір складається з елементів {A, B, C}, то правило $A \Rightarrow B$ не повинно розглядатися.

Залежно від розміру щонайдовшого часто зустрічающегося набору, алгоритм Apriori сканує базу даних певну кількість разів. Різновиди алгоритму Apriori, що є його оптимізацією, запропоновані для скорочення кількості сканувань бази даних, кількості наборів-кандидатів або того і іншого. Були запропоновані наступні різновиди алгоритму Apriori: AprioriTid і AprioriHybrid.

Алгоритм AprioriTid. Цей алгоритм генерує набори елементів використовуючи лише великі набори, знайдені на попередньому кроці, без повторного розгляду транзакцій. AprioriTid покращує Apriori за рахунок того, що використовує базу даних лише при першому проході. При підрахунках на подальших кроках використовуються лише дані, створені при першому проході і такі, що мають набагато менший розмір, чим вихідна база даних. Це приводить до колосального зростання продуктивності - в три рази швидше, ніж AIS і в чотири рази швидше, ніж SETM.

Алгоритм AprioriHybrid. Аналіз часу роботи алгоритмів Apriori і AprioriTid показує, що в ранішніх проходах Apriori досягає більшого успіху, чим AprioriTid; проте AprioriTid працює краще Apriori в пізніших проходах. Крім того, вони використовують одну і ту ж процедуру формування наборів-кандидатів. Заснований на цьому спостереженні, алгоритм AprioriHybrid запропонований, аби об'єднати кращі властивості алгоритмів Apriori і AprioriTid. AprioriHybrid використовує алгоритм Apriori в початкових проходах і переходить до алгоритму AprioriTid, коли очікується, що закодований набір первинної множини в кінці проходу відповідатиме можливостям пам'яті. Проте, перемикання від Apriori до AprioriTid вимагає залучення додаткових ресурсів.

Алгоритм DHP. Проблема Apriori в тому, що він генерує надто багато двоелементних наборів, які не є частими. J. Park, M. Chen and P. Yu був запропонований механізм *прямого хешування і обрізання даних* (direct hashing and pruning, DHP), які дозволяють зменшити кількість наборів кандидатів за рахунок відкидання k наборів з хеш-таблиці, якщо їх значення не досягає величини мінімальної підтримки. В основі роботи алгоритму - імовірнісний підрахунок наборів-кандидатів, здійснюваний для скорочення числа

підраховуваних кандидатів на кожному етапі виконання алгоритму Apriori. Скорочення забезпечується за рахунок того, що кожен з k-елементних наборів-кандидатів окрім кроку скорочення проходить крок хешування. У алгоритмі на $k-1$ етапі під час вибору кандидата створюється хеш-таблиця. Кожен запис хеш-таблиці є лічильником всієї підтримки k-елементних наборів, які відповідають цьому запису в хеш-таблиці. Алгоритм використовує цю інформацію на етапі k для скорочення множини k-елементних наборів-кандидатів. Після скорочення підмножини, як це відбувається в Apriori, алгоритм може видалити набір-кандидат, якщо його значення в хеш-таблиці менше порогового значення, встановленого для забезпечення. Як показали порівняльні експерименти, ця чудова фільтруюча особливість дозволяє DHP завершити вже всі обчислення в тому момент, коли Apriori все ще знаходиться на другому кроці.

Подальші зусилля по поліпшенню алгоритму Apriori пов'язані з розпаралелюванням алгоритму. Було запропоновано 3 паралельні алгоритми, що дозволяють прискорити пошук часто використовуваних наборів елементів. Алгоритм *Count Distribution* (CD) мінімізує обмін даними (communication) за рахунок проведення дублюючих обчислень. Алгоритм *Data Distribution* (DD) використовує основну пам'ять системи для передачі локальних даних на інші вузли системи. Алгоритм *Candidate Distribution* - це збалансований по навантаженню алгоритм, що знижує синхронізацію між процесорами і сегментує базу даних на основі різних шаблонів транзакцій. Ці паралельні алгоритми були протестовані і алгоритм CD мав найкращу продуктивність в порівнянні з алгоритмом Apriori. Його накладні витрати не перевищували 7.5% в порівнянні з Apriori.

Масштабованість - інша важлива область дослідження асоціативних алгоритмів, оскільки бази даних з кожним днем стають все більше і більше. Алгоритми мають бути здатні масштабуватися, аби обробити великі набори даних. Ґрунтуючись на цій ідеї була зроблена спроба зробити DD і CD алгоритми масштабованими. Були, відповідно, створені алгоритми *Intelligent*

Data Distribution (IDD) і *Hybrid Distribution* (HD). IDD направлений на зниження накладних витрат, а також на взаємодію і виключення зайвих обчислень за рахунок використання агрегатної пам'яті (aggregate memory) для розділення кандидатів і ефективного переміщення даних. HD був покращен в порівнянні з IDD за рахунок динамічного розділення масиву кандидатів для кращої підтримки збалансованого навантаження. Експерименти показали, що час відгуку IDD в 4.4 разу менше, ніж DD на 32-процесорній системі, а HD виявився кращим CD на 9.5% при роботі на 128-процесорах.

Інше дослідження масштабування засновано на введенні *полегшених структур даних* Segment Support Map (SSM), які зменшують кількість наборів-кандадитів, необхідних в розрахунках. SSM містить значення підтримки для одинарних наборів. Шляхом підсумовування значень підтримки для одинарних наборів оцінюються верхні кордони для k-елементних наборів. Стосовно Apriori, зусилля по генеруванню одноелементних наборів можуть бути заощаджені простим пошуком тих одноелементних наборів в SSM, в яких підтримка вище мінімального порогу. Більш того, одноелементні набори, в яких підтримка менше порогового значення, взагалі виключаються, що дозволяє понизити кількість наборів більш високого рівня.

Інша область досліджень, що направлена на поліпшення Apriori і базується на використанні оригінальних структур даних, зв'язана із застосуванням *дерев часто зустрічающихся елементів* (frequent pattern tree, or FP-tree). FP-дерево зберігає інформацію про часто використовувані шаблони (patterns). Метод, заснований на FP-деревах, називається *FP-growth* (метод вирощування найбільш популярних шаблонів). У ньому, замість генерації наборів-кандидатів, пропонується шукати часто використовувані шаблони. FP-growth метод на порядок кращий ніж Apriori і також краще масштабується.

Перехід від ітераційних методів, подібно Apriori і DHP, до інноваційного використання структур даних, типа SSM і FP-дерев, привів до того, що пошук часто зустрічающихся наборів елементів став більш масштабованим і ефективним. Необхідність в швидших і більш масштабованих

алгоритмах як і раніше існує, оскільки бази даних стають все більше і більше з кожним днем. Дослідження, присвячені розподіленим алгоритмам для пошуку наборів, що часто зустрічаються, потребують тим більше уваги, чим більше баз даних інтегрується разом. Така інтеграція переводить задачу пошуку на новий рівень, що вимагає гнучкіших алгоритмів, які беруть до уваги різне представлення однакових даних. Наприклад, zip-коди можуть бути представлені рядками або числами, тому дані не можуть бути нормалізовані до проведення пошуку. Можна чекати збільшення кількості досліджень, пов'язаних з паралельними алгоритмами, оскільки мережеві обчислення набирають все більшу популярність.

Дослідження, присвячені генерації правил, в основному фокусуються на нових алгоритмах (таких, що дають більше типів асоціативних правил) і цікавості правил. Нові алгоритми в основному використовують нові стратегії, такі, як *паралельні обчислення* і *Evolutionary Algorithm* (EA). Нові типи правил додають розмірність і якість в традиційні булеві типи правил. Приклади - кількісні асоціативні правила і часові асоціативні правила. Нові критерії цікавості мають тенденцію бути об'єктивнішими, ніж підтримка і достовірність. Більшість асоціативних правил згенерували шляхом підрахунку кількості транзакцій, в яких правило зустрічається в базі даних - достовірності. Інтуїтивно ясно, що набори часто зустрічающихся елементів можна розділити на частини і проводити підрахунки в паралель.

Останніми роками, алгоритм EA був широко схвалений в багатьох наукових областях. EA запозичує механізми біологічної еволюції і застосовує їх для вирішення проблем. Особливо добре він використовується для пошукових і оптимізаційних проблем, так що проблема пошуку асоціативних правил якраз підходить для вирішення цим алгоритмом. В комп’ютерних системах інтелектуального аналізу даних EA був використаний для генерації асоціативних правил. Популяція наборів, що часто зустрічаються, бралася як початкова популяція. Використання EA включає пересічення і мутацію цих наборів. Популяція еволюціонує так, що в ній залишаються лише набори, що

щонайкраще задовольняють функції придатності (fitness function). Коли в популяції залишається бажана кількість часто зустрічающихся наборів, алгоритм зупиняється. Це оригінальна дорога генерування та пошуку асоціативних правил застосовується для пересічних інтервалів в різних наборах. Наприклад, один набір може мати інтервал [10, 20], а інший [15, 25]. Тут є разючий контраст в порівнянні з іншими технологіями, які розділяють атрибути на інтервали, що не перетинаються. Правила, які потрапляють в пересічення двох інтервалів можуть бути не знайдені із-за втрати інформації. Алгоритм EA дозволяє знайти нові правила.

При розробці сучасних систем data mining були проведені також дослідження відносно типів асоціативних правил. На першому етапі еволюції таких систем домінували булеві асоціативні правила. Пізніше, фокус змістився на кількісні асоціативні правила. Кількісні асоціативні правила - це правила над кількісними і категоріальними атрибутами, подібно зросту і сімейному стану. Пошук таких правил включає розділення діапазону значень атрибутів на частини, при цьому інформація при такому розділенні може втрачатися. Був запропонований алгоритм пошуку кількісних асоціативних правил, заснований на Apriori. У ньому вводиться поняття часткової завершеності (partial completeness) для вимірювання втрат інформації при розділенні і інтерес, «більш ніж очікуваний» («greater than expected» interest) як міра цікавості. Часткова завершеність прямо пропорційна втратам інформації. Набуваючи мінімального значення підтримки і часткової завершеності від користувача, система може визначити необхідну кількість розділень. Кількісне правило цікаве лише, якщо воно має «більш ніж очікувану» підтримку і достовірність, вказану користувачем.

Розмірність часу одна з тих характеристик, що існує для всіх транзакцій. Тому, вона має бути включена в процес пошуку наборів, що часто зустрічаються, особливо, коли не всі елементи існують протягом всього періоду, який охоплюють зібрані дані. Існує концепція часу шляхом обмеження пошуку часто зустрічающихся наборів згідно часу життя елементів. Також

введена концепція тимчасової підтримки на додаток до звичайної підтримки і достовірності. Час життя елементу визначається як перший і останній момент часу появи елементу в базі даних. Тимчасова підтримка - це ширина мінімального інтервалу. Таким чином, правило розглядається в тому випадку, якщо воно має досить високі значення підтримки і тимчасової підтримки. Побічним продуктом даного підходу є те, що старі або застарілі набори елементів, можуть бути відкинуті.

Будь-яка асоціація, знайдена вказаними вище алгоритмами, може бути правилом. Якість цих правил вимірюється достовірністю. Проте, лише ті правила, достовірність яких вище певного рівня, цікаві і заслуговують на увагу. Більшість алгоритмів визначають цікавість в термінах величин підтримки і достовірності, вказаних користувачем. Проблема в тому, що ці алгоритми покладаються на те, що користувачі зможуть задати відповідні значення. Новий алгоритм, названий APACS2, не вимагає від користувача яких-небудь припущень, але використовує об'єктивно цікаві значення, названі регульованою різницею (adjusted difference). Більш того, APACS2 може виявляти як позитивні, так і негативні асоціативні правила. Також представлена нова концепція зв'язаності (relatedness) як альтернативного підходу для визначення цікавості. APACS2 використовує регульовану різницю як об'єктивну міру цікавості. Регульована різниця визначається в термінах стандартизованої різниці (standardized difference) і оцінки максимальної правдоподібності (maximum likelihood estimate). Якщо величина регульованої різниці вища, ніж 1.96, тобто 95% нормального розподілу, асоціація розглядається як істотно різна і, отже, цікава. Якщо регульована різниця позитивна, це означає, що правило ймовірно, і навпаки. Направлена природа регульованої різниці дає нову розмірність пошуку асоціативних правил.

Цікавість може бути суб'єктивною, якщо застосовуються значення підтримки і достовірності, або об'єктивною, якщо використовується регульована різниця. Протилежна до концепції - зв'язність (relatedness), - була введена для того, щоб досліджувати співвідношення між двома елементами, на

основі частоти їх спільної появи в транзакціях і контексту. Зв'язність призначена для використання замість цікавості для кількісних правил. В зв'язності є три компоненти: середня передбачаюча здатність присутності одного елементу при присутності іншого, інтенсивність появи пар елементів у присутності інших елементів; заміщаємість іншого елементу для елементу в парі елементів. Ці три заходи складають силу зв'язності в термінах частоти правила по відношенню до інших елементів.

Дослідження генерації правил починалося з простих булевих типів з використанням суб'єктивних значень підтримки і достовірності. Нині воно включає різних типів правил з об'єктивними характеристиками, подібно до регульованих різниць і зв'язності. Нові дослідження додали кількісні і якісні аспекту генерації правил. Якість покращена за рахунок використання об'єктивних властивостей якості правил. Кількість збільшена за рахунок вживання оригінальних методологій, що дозволяють здійснювати пошук правил в пересічних інтервалах і знаходити негативні асоціації. З появою все нових і нових типів даних, наприклад, мультимедійних даних, більше досліджень стало проводитися для визначення нових типів асоціативних правил. Це дозволить проаналізувати нові шаблони поведінки і зробити нові передбачення. Здатність перетворювати мультимедійні дані в raw-формат (формат необроблених даних) може знайти практичне застосування в медицині, національній безпеці та інших областях.

Технології пошуку асоціативних правил вивчається и застосовуються понад двадцять років. Безліч фундаментальних досліджень вже проведена. Велику увагу було сфокусовано на продуктивності і масштабованості алгоритмів, але недостатньо уваги було приділено якості (цікавості) правил, що генерувалися. У наступні десятиліття, увага сфокусується на практичному впровадженні досліджень в різних сферах нашого життя, наприклад, генетичних дослідженнях, медицині, національній безпеці і тому подібне. У міру інтеграції баз даних і із збільшенням в них наборів даних, пошуку алгоритмів, що дозволяють сканувати більше і швидше, приділятиметься

менше увага. Навпаки, розподіленим алгоритмам, що дозволяють розділяти навантаження в обчислювальних мережах, уваги буде приділятися все більше. Чим більше даних створюється поза традиційними базами даних, data mining і пошук правил переростуть сканування таблиць баз даних і почнуть працювати з даними в raw-форматі, наприклад, з відеокліпами. Питання продуктивності і масштабованості стануть ще важливіші. Нові типи даних можуть бути необхідні для просування нових типів аналізу даних. Так само стають більш потрібнішими об'єктивні заходи цікавості, аби експерти предметних областей могли уникнути маніпуляцій з критеріями пошуку правил при здобутті бажаних результатів. Пошук асоціативних правил залишатиметься благодатною темою для досліджень. На основі досліджень і розвитку в останнє десятиліття, в ньому сформувалися і виділилися області, які повинні колосально змінитися в найближче десятиліття.

4.2. Програмні засоби пошуку асоціативних правил.

За допомогою алгоритмів виявлення асоціативних правил можна вирішувати чималий спектр практичних завдань. Саме тому ринок програмних продуктів, що реалізовують ці технології, досить представницький та різнорідний. Практично кожна відома компанія в тому або іншому вигляді використовує технології пошуку асоціативних правил в своїх програмних продуктах. Зупинимося на деяких програмних рішеннях, що отримали найбільшу популярність [2, 25, 70, 77].

Пакет Deductor. Розглянемо приклад рішення задачі пошуку асоціативних правил. Вважатимемо, що існує транзакційна база даних. Необхідно знайти набори товарів, що зустрічаються найбільш часто, і набір асоціативних правил з певними границями значень підтримки і довіри.

Процес побудови асоціативних правил виконаємо в аналітичному пакеті Deductor. Транзакційна база даних, яка містить в кожній записі номер чека і

товар, придбаний по цьому чеку, має формат MS Excel. Спершу імпортуємо дані з файлу MS Excel в середовище Deductor. Для номера транзакції (зазвичай в базі даних - це поле «номер чека») вказуємо тип «ідентифікатор транзакції (ID)», а для найменувань товару - тип «елемент». Результат імпорту бази даних з файлу MS Excel в середовище Deductor відображені на рисунку 4.6.

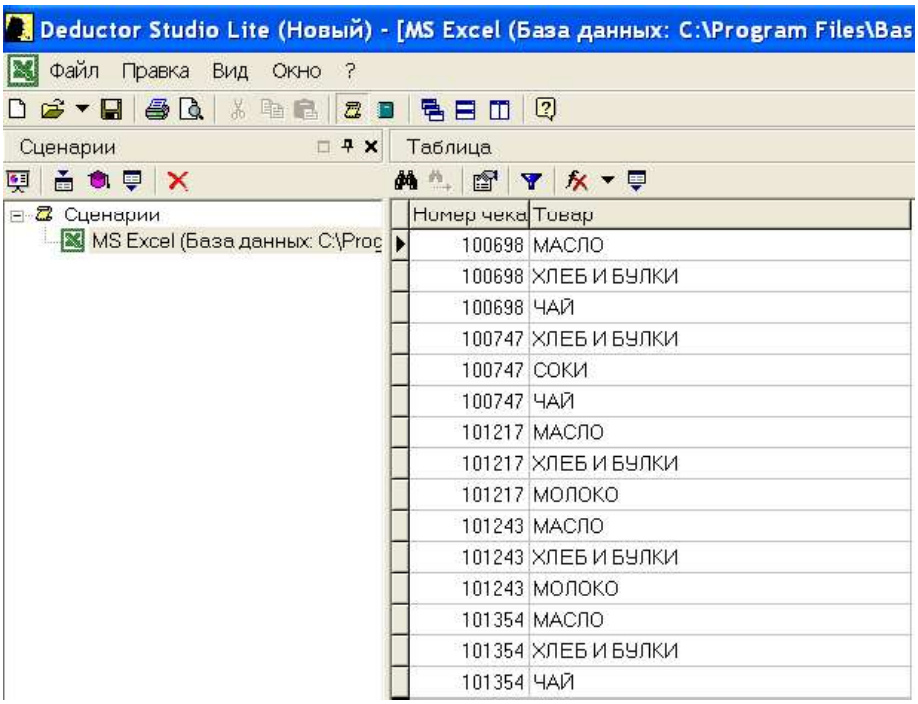


Рис. 4.6. Транзакційна база даних, яка імпортована з файлу MS Excel.

За допомогою майстра обробки вибираємо метод «Асоціативні правила». На другому кроці майстра перевіряємо призначення вихідних стовпців даних, вони повинні мати тип «ID» і «елемент». На третьому кроці, проілюстрованому на рис. 4.7, необхідно настроїти параметри пошуку правил, тобто встановити мінімальні і максимальні характеристики підтримки і достовірності. Це найбільш «відповідальний» момент формування набору правил. Вибір можна зробити на основі яких-небудь міркувань, наявного досвіду аналізу подібних даних, інтуїції або ж визначити в ході експериментів.

Встановимо наступні границі для параметрів пошуку: мінімальний і максимальний рівень підтримки дорівнюють 20% і 60% відповідно, мінімальний і максимальний рівень значення достовірності дорівнюють 40% і

90% відповідно. Ці значення були виявлені в ході проведення декількох експериментів, і виявилось, що саме при таких значеннях формується необхідний набір правил. При застосуванні деяких інших значень, наприклад, рівня підтримки від 30% до 50%, набір правил не формується, оскільки жодне правило по параметрах підтримки не входить в цей інтервал.

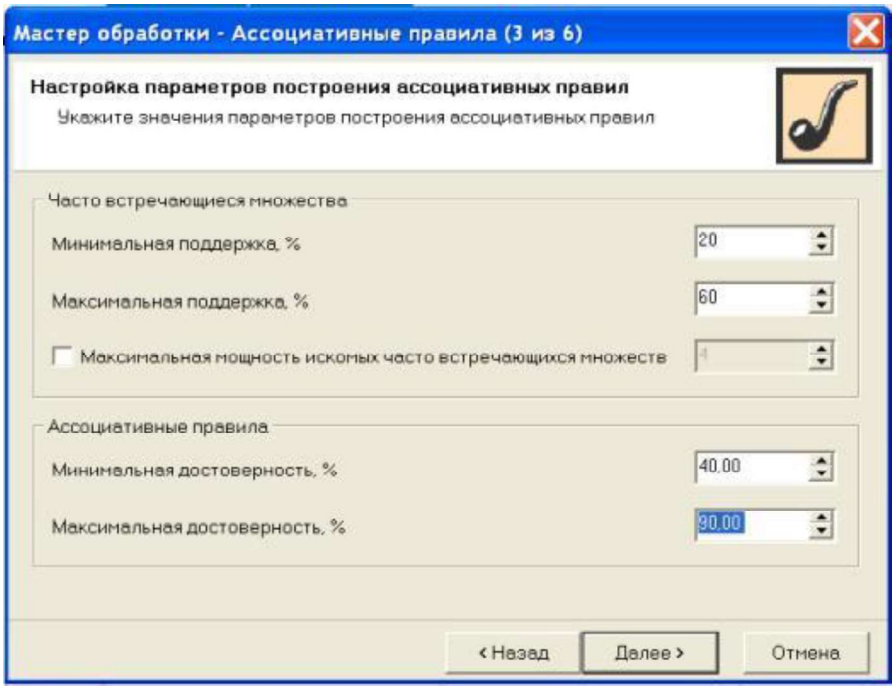


Рис. 4.7. Налаштування параметрів побудови асоціативних правил.

На наступному кроці майстра запускається процес пошуку асоціативних правил. В результаті бачимо інформацію про кількість множин і знайдених правил у вигляді гістограми розподілу часто зустрічающихся множин по їх потужності. Даний процес проілюстрований на рис. 4.8.

Ми бачимо, що кількість сформованих множин дорівнює тринадцяти - це популярні набори, кількість сформованих правил - п'ятнадцять. На наступному кроці для перегляду отриманих результатів пропонується вибрати візуалізацію із списку. Виберемо такі: «Популярні набори», «Правила», «Дерево правил», «Що-якщо».

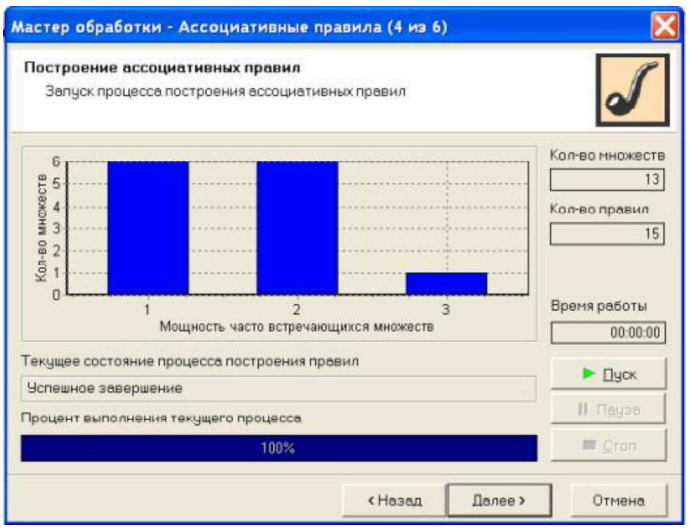


Рис. 4.8. Процес побудови асоціативних правил.

Візуалізація «Популярні набори». Популярні набори або набори, що часто зустрічаються, - це набори, що складаються з одного або декількох товарів, які в транзакціях найчастіше зустрічаються одночасно. Характеристикою, наскільки часто набір зустрічається в аналізованому наборі даних, є підтримка. Популярні набори нашого набору даних, знайдені при заданих параметрах, приведені на рис 4.9. Є можливість відсортувати дану таблицю згідно різним її характеристикам. Для визначення найбільш популярних товарів і їх наборів зручно відсортувати її по рівню підтримки. Таким чином, ми бачимо, що найбільшою популярністю користуються такі товари: хліб і булки, масло, соки.

N	Множество	↑Поддержка	
		%	Кол-во
6	ХЛЕБ И БУЛКИ	54,55	24
3	МАСЛО	52,27	23
5	СОКИ	50,00	22
10	МАСЛО **И** ХЛЕБ И БУЛКИ	45,45	20
4	МОЛОКО	43,18	19
2	КЕФИР	31,82	14
1	ЙОГУРТЫ	31,82	14
12	СОКИ **И** ХЛЕБ И БУЛКИ	22,73	10
11	МОЛОКО **И** ХЛЕБ И БУЛКИ	22,73	10
8	МАСЛО **И** МОЛОКО	22,73	10

7	ЙОГУРТЫ **И** КЕФИР	22,73	10
13	МАСЛО **И** МОЛОКО **И** ХЛЕБ И БУЛКИ	20,45	9
9	МАСЛО **И** СОКИ	20,45	9

Рис. 4.9. Візуалізація «Популярні набори».

Візуалізація «Правила». Правила в даній візуалізації розміщені у вигляді списку. Кожне правило, представлене як «умова-наслідок», характеризується значенням підтримки в абсолютному і процентному вираженні, а також достовірністю. Таким чином, аналітик бачить поведінку покупців, описану у вигляді набору правил. Набор правил для вирішуваної нами задачі приведений на рис. 4.10. Наприклад, перше правило говорить про те, що якщо споживач купив йогурт, то з достовірністю або вірогідністю 71% він купить також кефір. Ця інформація корисна з різних точок зору. Вона, наприклад, допомагає вирішити завдання розташування товарів в магазині.

N	**Условие**	**Следствие**	**Поддержка**		**Достоверность, %**
			%	**Кол-во**	
1	ЙОГУРТЫ	КЕФИР	22,73	10	71,43
2	КЕФИР	ЙОГУРТЫ	22,73	10	71,43
3	МАСЛО	МОЛОКО	22,73	10	43,48
4	МОЛОКО	МАСЛО	22,73	10	52,63
5	СОКИ	МАСЛО	20,45	9	40,91
6	МАСЛО	ХЛЕБ И БУЛКИ	45,45	20	86,96
7	ХЛЕБ И БУЛКИ	МАСЛО	45,45	20	83,33
8	МОЛОКО	ХЛЕБ И БУЛКИ	22,73	10	52,63
9	ХЛЕБ И БУЛКИ	МОЛОКО	22,73	10	41,67
10	СОКИ	ХЛЕБ И БУЛКИ	22,73	10	45,45
11	ХЛЕБ И БУЛКИ	СОКИ	22,73	10	41,67
12	МАСЛО **И** МОЛОКО	ХЛЕБ И БУЛКИ	20,45	9	90,00
13	МАСЛО **И** ХЛЕБ И БУЛКИ	МОЛОКО	20,45	9	45,00
14	МОЛОКО **И** ХЛЕБ И БУЛКИ	МАСЛО	20,45	9	90,00
15	МОЛОКО	МАСЛО **И** ХЛЕБ И БУЛКИ	20,45	9	47,37

Рис. 4.10. Візуалізація «Правила».

При великій кількості знайдених правил і широкому асортименті товарів аналізувати отримані правила досить складний. Для зручності аналізу таких наборів правил пропонується візуалізація «Дерево правил» і «Що-якщо».

Візуалізація «Дерево правил» є дворівневим деревом, яке може бути побудоване по двох критеріях: по умові і по наслідку. Якщо дерево побудоване по умові, то зверху списку відображується умова правила, а список, що додається до даної умови, складається з його наслідків. При виборі певної умови, в правій частині візуалізації відображуються наслідки умови, рівень підтримки і достовірності. В разі побудови дерева по наслідку, зверху списку відображується наслідок правила, а список складається з його умов. При виборі певного наслідку, в правій частині візуалізації ми бачимо умови цього правила з вказівкою рівня підтримки і достовірності.

Візуалізація «Що-якщо» зручний, якщо нам необхідно відповісти на питання, які наслідки можуть вийти з даної умови. Наприклад, вибравши умову «МОЛОКО», в лівій частині екрану отримуємо три наслідки «МАСЛО», «ХЛІБ І БУЛКИ, «МАСЛО І ХЛІБ І БУЛКИ», для яких вказані рівень підтримки і достовірності. Ця візуалізація представлена на рис. 4.11.

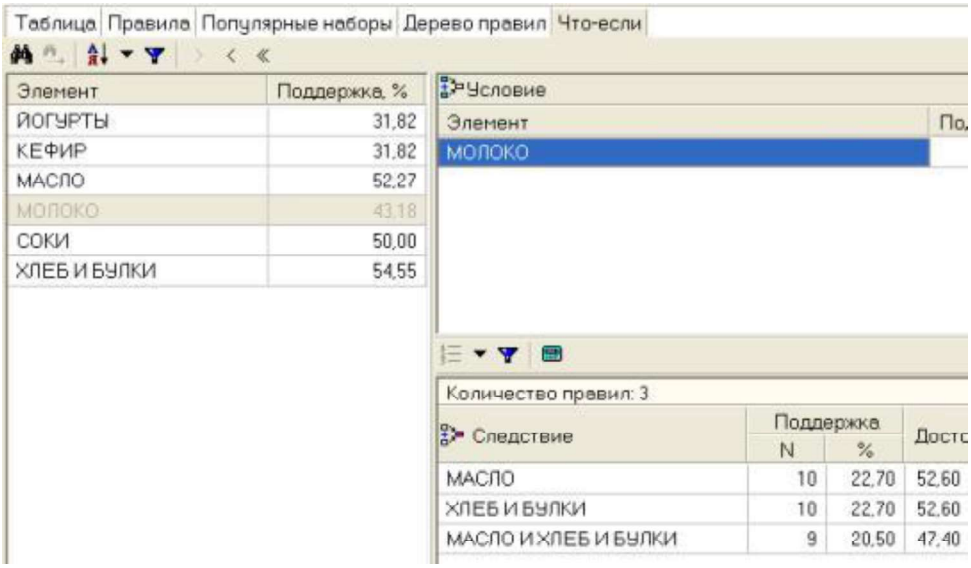


Рис. 4.11. Візуалізація «Що-якщо».

Система PolyAnalyst. Система PolyAnalyst призначена для автоматичного аналізу числових і текстових даних з метою виявлення в них раніше невідомих, нетривіальних, практично корисних і доступних розумінню закономірностей, необхідних для ухвалення оптимальних рішень в бізнесі і в інших областях людської діяльності. В даний час вона є однією з найпотужніших систем Data Mining в світі, реалізованих для Intel платформ і операційних систем Microsoft Windows. Аналогічні системи Data Mining таких провідних виробників, як IBM (Intelligent Miner, Data Miner), Silicon Graphics (SGI Miner), Integral Solutions (Clementine), SAS Institute (SAS) працюють на середніх і великих машинах і коштують десятки і навіть сотні тисяч доларів. Завдяки унікальній технології «Еволюційного програмування» та іншим інноваційним математичним алгоритмам, PolyAnalyst поєднує в собі високу продуктивність «великих систем» з низькою вартістю, властивою програмам для Windows. PolyAnalyst набув широкого поширення в світі. Більше 500 інсталяцій в 20 країнах світу, серед користувачів системи значний список складають найбільші світові корпорації: Boeing, 3M, Chase Manhattan Bank, Dupont, Siemens та інші. Розглянемо реалізацію технологій пошуку асоціативних правил в цьому програмному комплексі.

Модуль Market Basket Analysis (BA) - метод аналізу «корзини покупця». Назва цього методу походить від задачі визначення які товари ймовірно купуються спільно. Проте реальна сфера його застосування значно ширша. Наприклад, продуктами можна вважати сторінки в Інтернеті, або ті або інші характеристики клієнта або відповіді респондентів в соціологічних і маркетингових дослідженнях та інше. Алгоритм BA отримує на вхід бінарну матрицю, в якій рядок - це одна корзина (наприклад, касовий чек), а стовпці заповнені логічними 0 і 1, що позначають наявність або відсутність даної ознаки (товару). На виході формуються кластери ознак, що спільно зустрічаються, з оцінкою їх вірогідності і достовірності. Окрім цього формуються асоціативні направлені правила типа: якщо ознака «A», то з деякою вірогідністю ще і ознака «B» і ще ознака «C». Алгоритм BA в

PolyAnalyst працює виключно швидко і здатний обробляти величезні масиви даних (рис. 4.12).

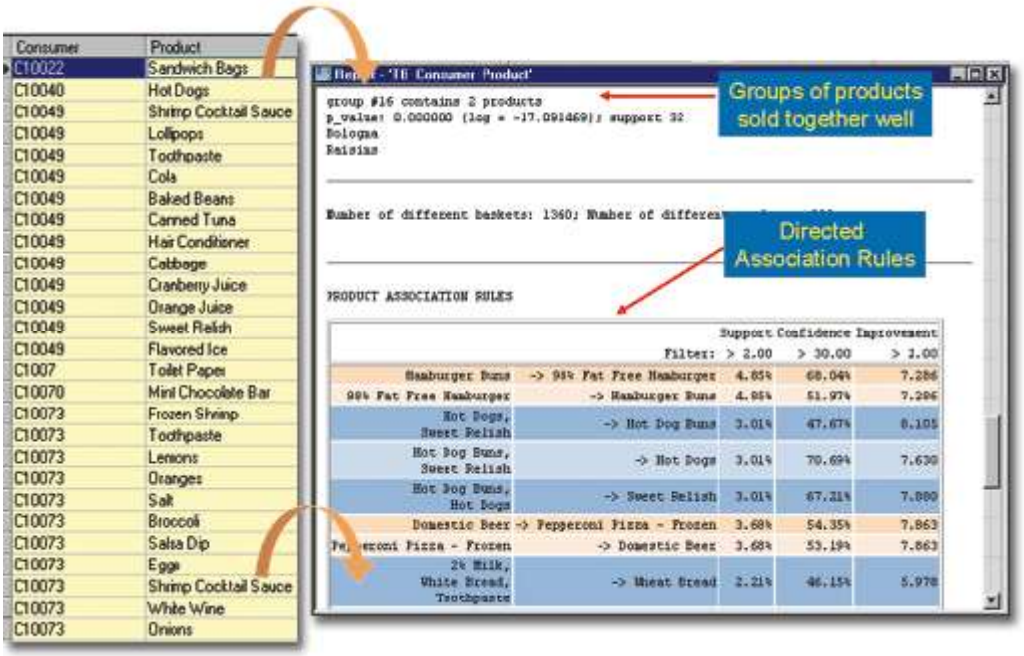


Рис. 4.12. Приклад роботи Market Basket Analysis.

Модуль Transactional Basket Analysis (ТБ) - транзакційний аналіз «корзини». Transactional Basket Analysis - це модифікація алгоритму BA, застосована для аналізу дуже великих даних, що не рідкість для цього типа задач. Він передбачає, що кожен запис в базі даних відповідає одній транзакції, а не одній корзині (набору куплених за одну операцію товарів). На основі цього алгоритму компанія «Мегапьютер» розробила окремий продукт - X-SellAnalyst, призначений для on-line рекомендації продуктів в Інтернет магазинах.

Однією з унікальних особливостей PolyAnalyst є інтеграція інструментів Data Mining - засобів аналізу числової інформації з методами аналізу текстів на природній мові - алгоритмів Text Mining.

Модуль Text Analysis (TA) - текстовий аналіз. Text Analysis є засобом формалізації неструктурованих текстових полів в базах даних. При цьому текстове поле представляється як набір булевих ознак, заснованих на наявності і частоті даного слова, стійкої словосполуки або поняття в даному тексті. При

цьому з'являється можливість розповсюдити на текстові поля всю потужність алгоритмів, реалізованих в системі PolyAnalyst. Крім того, цей метод може бути використаний для кращого розуміння текстової компоненти даних за рахунок автоматичного виділення найбільш ключових понять (рис. 4.13).

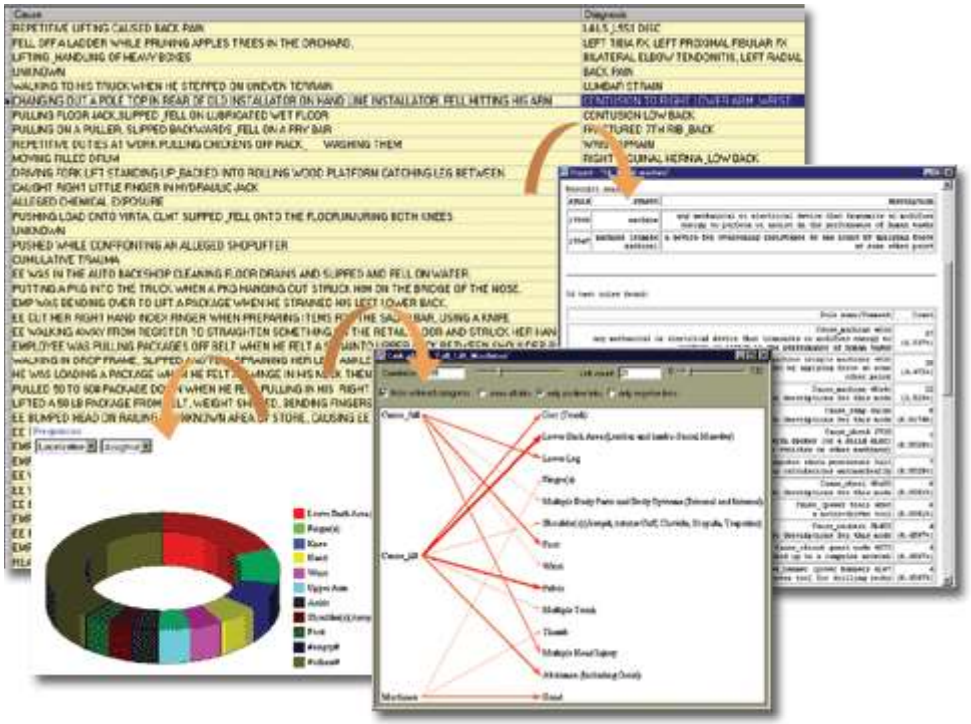


Рис. 4.13. Текстовий аналіз на основі асоціацій.

Модуль Text Categorizer (TC) - каталогізатор текстів. Цей модуль дозволяє автоматично створити ієрархічний деревовидний каталог наявних текстів і помітити кожен вузол цієї деревовидної структури найбільш індикативною ознакою для текстів, що відносяться до нього. Це потрібно для розуміння тематичної структури аналізованої сукупності текстових полів і ефективної навігації по ній.

Модуль Link Terms (LT) - зв'язок понять. Цей модуль дозволяє виявляти зв'язки між поняттями, що зустрічаються в текстових полях бази даних і представляти їх у вигляді графа. Цей граф також може бути використаний для виділення записів, що реалізовують вибраний зв'язок (рис. 4.14).

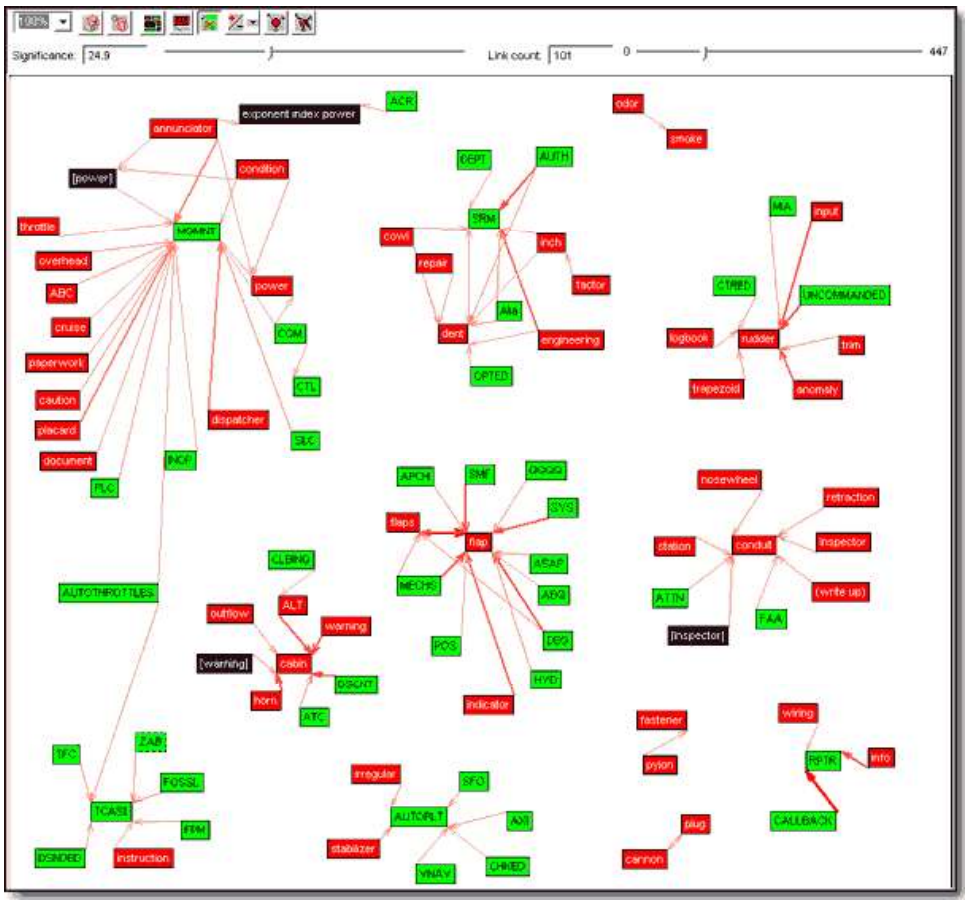


Рис. 4.14. Результати роботи модуля Link Terms.

Пакет STATISTICA. Виконання алгоритму Apriori в пакеті STATISTICA дозволяє швидко обробляти великі бази даних для зв'язків, заснованих на зумовлених «відправних пунктах» значень для дослідження. Корисність цього методу при вирішенні нестандартних задач видобутку даних краще всього розглянути на прикладі. Будемо вважати, що збираються дані з апарату, реєструючого готівку у великому книжковому магазині. Інформація по кожному покупцеві заноситься в базу даних, вона складається з назв куплених книг, додаткових назв магазина і інших куплених товарах. Отже, кожен запис в базі даних представляє одного покупця (транзакцію), і може складатися з однієї купленої книги або множини (можливо сотень) різних пунктів, розташованих в довільному порядку, залежно від послідовності їх поступлення і реєстрації пристроєм. Мета аналітиків – знайти зв'язки між купленими товарами, тобто виявити зв'язки між назвами товарів і частотою їх вжитку. Наприклад, Ви

хочете взнати, яку ще книгу захоче придбати покупець, що вже купив одну. Ця інформація може бути швидко використана для пропозиції покупцеві якихось додаткових найменувань товарів (рис. 4.15).

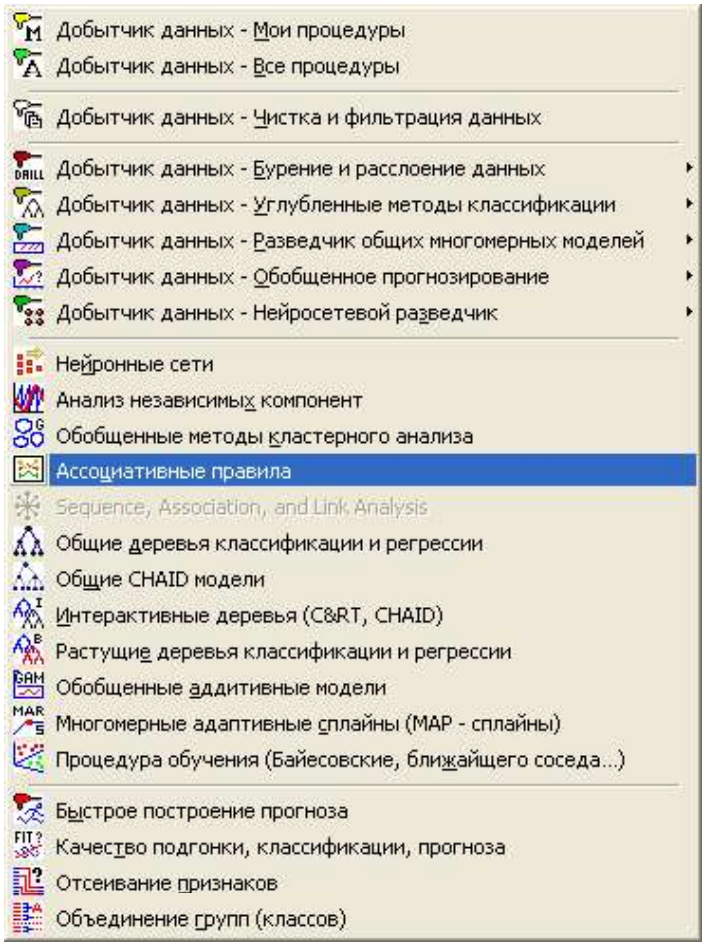


Рис. 4.15. Меню Data Mining пакету STATISTICA.

Правила зв'язків в STATISTICA підтримують всі основні типи даних і формати, в яких зазвичай записуються категорії, об'єкти або протоколи (наприклад, інформація про покупку).

Змінні багатовимірного відгуку. Змінні багатовимірного відгуку зазвичай містять багатовимірні змінні (точніше, список змінних) які можуть містити, для кожного результату спостереження окремо, кодові або текстові значення, що описують окремий «вимір» або транзакцію. Прикладом використання змінних багатовимірного відгуку є: продавець зафіксував

покупку споживача в окремому записі, де кожен запис може містити одне або більш придбань, причому в довільному порядку. Це найприродніший формат даних для зберігання інформації про покупки споживача.

Багатовимірні дихотомії. У цьому форматі даних кожна змінна представлена однією подією або категорією, і дані дихотомії в кожній змінній відображають так чи інакше відповідний об'єкт або категорію, що відноситься до певного спостереження. Наприклад, передбачимо що продавець створив таблицю даних, в якій кожен стовпець містить товари, які можна купити. Кожна транзакція (ряд таблиці даних) записуватиме купив чи ні відповідний покупець цей товар, тобто задіяна чи ні відповідна транзакція.

Первинна обробка даних: підтримка. В першу чергу STATISTICA сканує всі змінні, аби визначити унікальні кодові або текстові значення, знайдені серед змінних для аналізу. При цій первинній обробці відповідна частота, з якою унікальні кодові або текстові значення зустрічаються в кожній транзакції, також будуть обчислені. Можливість того, що транзакція містить певне кодове або текстове значення називається підтримкою. Підтримка також обчислюється при подальших послідовних обробках даних, як імовірність зустрічі подвійних, потрійних і так далі кодових або текстових значень (об'єктах), тобто визначається окремо для «Причини» і «Наслідок» кожного зв'язку.

Вторинна обробка даних: довіра, кореляція. Після первинної обробки даних, всі об'єкти, в яких значення підтримки менше, ніж деякий визначений заздалегідь мінімум підтримки, будуть збережені в пам'яті для подальших обробок даних. Особливістю є те, що STATISTICA обчислюватиме умовну ймовірність для всіх пар кодових і текстових значень, в яких значення підтримки більше, ніж деякий певний мінімум підтримки. Ця умовна ймовірність - результат, який містить кодове або текстове значення X також містить кодове або текстове значення Y, - називається довіра.

На додаток STATISTICA обчислить підтримку для кожної пари кодових або текстових значень і кореляцію, засновану на підтримці. Значення кореляції

для пари кодових або текстових значень $\{X, Y\}$ обчислюється як підтримка цієї пари, що ділиться на квадратний корінь з величини підтримки X і Y. Після другої обробки даних програма збереже в пам'яті ті пари кодових або текстових значень, які:

- мають значення довіри, більше чим деякий визначений користувачем мінімум довіри;
- мають підтримку, більшу чим деякий визначений користувачем мінімум підтримки;
- мають значення кореляції, більше чим деяка мінімальна кореляція.

Подальші обробки даних: максимальний розмір об'єкту в «Причина», «Наслідок». STATISTICA продовжуватиме сканувати дані, обчислюючи підтримку, довіру і кореляцію для подвійних кодових або текстових значень, потрійних і так далі При кожному повторенні програма витягуватиме правила зв'язку вигляду: *якщо «Причина» то «Наслідок»*, де «Причина» і «Наслідок» представлені кодовими або текстовими значеннями (об'єктами), або комбінацією кодових або текстових значень (об'єктів). Процес продовжуватиме до тих пір, поки ще можуть бути знайдені зв'язки, що задовольняють мінімуму значення підтримки, довіри і умови кореляції. Процес може продовжувати вибудовувати дуже складні правила зв'язку, наприклад, якщо X_1 і X_2 … і X_{20} то Y_1 і Y_2 ... і Y_{20}. Аби уникнути небажаного ускладнення, користувач додатково може точно встановити максимальну кількість кодових або текстових значень (об'єктів) в правилах зв'язку «Причина» і «Наслідок». Тоді це значення сприйматиметься як максимальний розмір об'єкту в зв'язку «Причина» і «Наслідок».

Основними статистиками, що обчислюються для зв'язків є підтримка (відносна частота умови «Причина» або «Наслідок»), довіра (умовна ймовірність «Причини», визначеної «Наслідком»), і кореляція (підтримка для «Причина» і «Наслідок», ділена на квадратний корінь з результату підтримки для «Причина» і підтримки для «Наслідок»). Ці показники можна звести в електронну таблицю (рис. 4.16). Ця таблиця результатів показує, як зв'язки

можуть бути застосовані до задачі аналізу тексту. Підтримка, значення довіри і кореляції виражені у відсотках.

Data: Summary of association rules (Scene 1)*

Summary of association rules (Scene 1.sta)
Min. support = 5.0%, Min. confidence = 5.0%, Min. correlation = 5.0%
Max. size of body = 10, Max. size of head = 10

	Body	==>	Head	Support(%)	Confidence(%)	Correlation(%)
154	and, that	==>	like	6.94444	83.3333	91.28709
126	like	==>	and, that	6.94444	100.0000	91.28709
163	and, PAROLLES	==>	will	5.55556	80.0000	73.02967
148	will	==>	and, PAROLLES	5.55556	66.6667	73.02967
155	and, you	==>	your	5.55556	80.0000	67.61234
122	your	==>	and, virginity	5.55556	57.1429	67.61234
164	and, virginity	==>	your	5.55556	80.0000	67.61234
121	your	==>	and, you	5.55556	57.1429	67.61234
73	that	==>	like	6.94444	41.6667	64.54972
75	that	==>	and, like	6.94444	41.6667	64.54972
161	and, like	==>	that	6.94444	100.0000	64.54972

Рис. 4.16. Пошук асоціативних правил в пакеті STATISTICA.

Графічне представлення зв'язків. Приведемо результати аналізу даних з прикладу Fastfood.sta. Опитувані в дослідженні вказували 3 їх улюблені фаст-фуд блюда. Правила зв'язки, отримані виходячи з цих даних, показані на 2М графіці зв'язку (рис. 4.17).

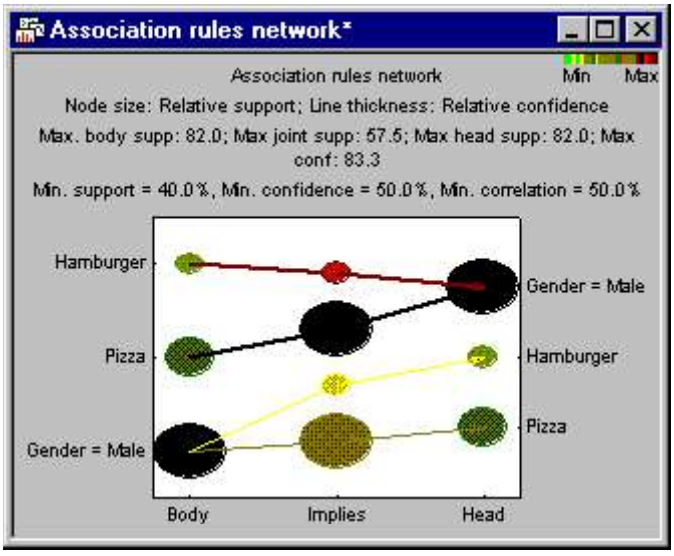


Рис. 4.17. Графічне відображення правила зв’язків.

Пункти, що визначають причини показані на графіці зліва, а наслідок - справа. Лінії, які сполучають причину із наслідком, відображують правила зв'язку. Значення підтримки для «Причина» і «Наслідок» для кожного зв'язку відбиваються в розмірі і кольорах кіл. Товщина кожної лінії відображає довірче значення (умовну ймовірність, яку для «Причина» визначає наслідок) для відповідного зв'язку; розміри і кольори кіл в центрі, над написом Implies відображають об'єднану підтримку (для тих, що зустрілися) відповідних компонентів «Причина» і «Наслідок».

У 3М графіці зв'язку підтримка для компонентів «Причина» і «Наслідок» кожного зв'язку відбиті в розмірі і кольорах кіл в 2М площині. Товщина кожної лінії відображає значення довіри (можливість об'єднання) для відповідного зв'язку; розміри і колір «плаваючих» кіл, побудованих напроти осі Z вказують на з'єднання підтримки (для частоти повторень) відповідних компонентів «Причина» і «Наслідок» правил зв'язку. Позиція накресленого кола по відношенню до осі Z відображає відповідне значення довіри. Отже, абсолютно точно визначений і виражений графічно висновок дозволяє сформулювати 2 правила: опитувані, які назвали Піца, як саме їх улюблене, також назвали Гамбургер, і навпаки Рис. 4.18).

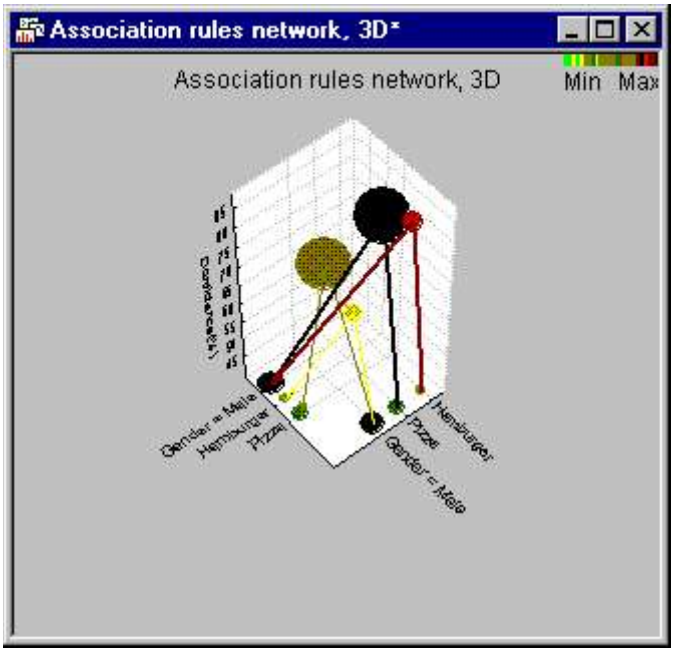


Рис.4.18. 3D відображення асоціативних правил в пакеті STATISTICA.

Система 1С: Предприятие 8.0. Одна з головних тенденцій на ринку обліково-управлінських систем – це постійне підвищення попиту на вживання засобів аналітичної обробки даних, що забезпечують ухвалення обґрунтованих управлінських рішень. Саме тому одним із стратегічних напрямів розвитку системи ПО «1С: Підприємство» є постійне розширення можливостей засобів економічної і аналітичної звітності. Підприємствам все частіше потрібні якісно інші засоби, що дозволяють автоматично шукати неочевидні правила і виявляти невідомі закономірності. Саме так можна генерувати якісно нові знання накопиченою компанією інформації і приймати деколи зовсім нетривіальні рішення для підвищення ефективності бізнесу, застосовуючи методи інтелектуального аналізу даних.

Пошук асоціацій Цей метод призначений для виявлення стійких комбінацій елементів в певних подіях або об'єктах. Результати аналізу представляються підсистемою у вигляді груп асоційованих елементів. Тут же окрім виявлених стійких комбінацій елементів наводиться розгорнута аналітика по асоційованих елементах (рис. 4.19).

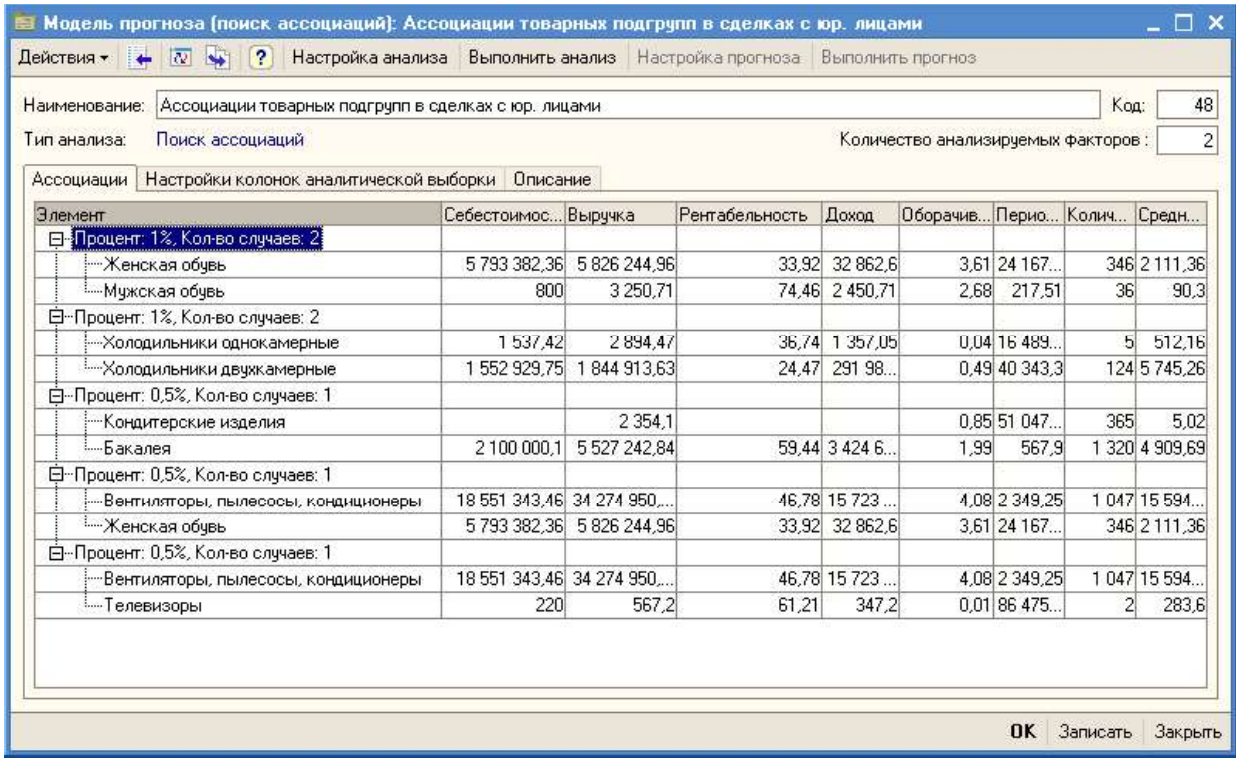


Рис. 4.19. Представлення результатів аналізу методом пошуку асоціацій".

Спочатку метод був розроблений для пошуку типових поєднань товарів в покупках, тому інколи його ще називають аналізом купівельної корзини. Стосовно цього сценарію в якості асоційованих елементів, як правило, виступають товарні групи або окремі товари. Групуючим об'єктом, що об'єднує елементи вибірок, може бути будь-який об'єкт інформаційної системи, що ідентифікує операцію, наприклад: замовлення покупця, акт про надання послуг або касовий чек.

Інформація про закономірності в товарних перевагах покупців дозволяє підвищити ефективність управління стосунками з клієнтами (у частині рекламних кампаній і маркетингових акцій), ціноутворення (формування комплексних пропозицій і системи знижок), управління запасами і мерчендайзинга (розподіл товарів в торгівельних залах). Іншим прикладом використання цього методу є визначення комбінацій рекламних каналів, яким віддаються переваги клієнтами, для виключення їх дублювання при проведенні цільових рекламних кампаній. Це дозволяє істотно понизити витрати на подібні заходи.

Реалізований в платформі алгоритм пошуку асоціацій має досить гнучкі засоби управління адекватністю моделей аналізу або прогнозу. Параметр «Мінімальний відсоток випадків» визначає «поріг спрацьовування» алгоритму на ту або іншу комбінацію елементів в події або об'єкті, що дозволяє не брати до уваги слабо поширені асоціації. Параметр «Мінімальна достовірність» визначає необхідну стійкість шукаємих асоціацій, а параметр «Мінімальна значущість» дозволяє виявити з них найбільш пріоритетні. Істотно полегшує сприйняття результатів аналізу і прогнозу параметр «Тип відсікання правил», який може набувати значень «Відсікати надлишкові» і «Відсікати покриті іншими правилами».

Для практичної інтерпретації результатів, отриманих даним алгоритмом критично важливе розбиття вихідної множини асоційованих елементів на дійсно однорідні з точки зору здійснюваного аналізу групи.

При створенні даної інформаційної системи були розроблені типові проекти застосування інтелектуального аналізу даних. Одним з таких проектів є «Управління ланцюжками постачань»

- *Сценарій* "Оптимізація вибору постачальників по товарній групі". Вибір домінуючих постачальників «першого ряду» для ключових товарних груп надзвичайно важливий для стабілізації системи логістики зокрема і загальної системи управління ланцюжками постачань в цілому, зменшення середньої тривалості ланцюжків постачань. З іншого боку, тісніша інтеграція з основними постачальниками дозволяє, як правило, істотно понизити собівартість товарів. У зв'язку з цим представляє інтерес аналіз стійких комбінацій постачальників в різних товарних групах порівняно з аналітикою по асоційованих в рамках груп постачальниках. Це дозволяє виявити «пересічення» постачальників в різних товарних групах і оптимізувати взаємини з ними.
- *Алгоритм* – «Пошук асоціацій».
- *Прогнозні атрибути* - стійкі комбінації постачальників.
- *Основні чинники* – товарні групи.
- *Розшифровка* - аналітика по постачальниках (об'єм закупівель, виручка, умови постачання, оплати, песимістичний, оптимістичний, середній терміни виконання замовлення).
- *Приклад закономірності*. Стійка асоціація крупного і непередбачуваного постачальника А і передбаченого середнього постачальника Б у великій кількості товарних груп. Можливо, при формуванні замовлень по конкурентних товарних групах як основного розглядати середнього постачальника, якщо обсяг замовлення великому не перевищує деякого (що дає істотний виграш на масштабах) порогу.

Клієнт Data Mining для Excel. Клієнт Data Mining для Excel дозволить провести повний цикл інтелектуального аналізу даних за допомогою клієнта Excel з використанням даних електронних таблиць або зовнішнього джерела, доступного базі даних Analysis Services, зокрема, пошук асоціативних правил (рис. 4.20).

Рис. 4.20. Панель інструментів клієнта Data Mining для Excel 2007.

Організація зліва направо та угрупування кнопок на панелі інструментів відображає типовий порядок виконання задач в проекті аналізу даних.

Підготовка даних. Задача вибору правильних атрибутів з джерела даних і представлення їх в правильному форматі займає високий відсоток часу в процесі побудови аналітичних моделей. Ця секція надає інструменти для основних задач підготовки даних до початку їх поглибленого аналізу.

Дослідження даних - служить для побудови графіка розподілу дискретних і безперервних змінних, а також для додавання угрупувань у вихідні дані.

Очищення даних - служить для видалення викидів і для зміни значень міток дискретних даних (наприклад, якщо вихідні дані містять значення «M» і «W» в колонці «Стать», а ви вважаєте за краще бачити ці значення як «чоловічий» і «жіночий» для ясності презентації результатів).

Секціонування даних - служить для розбиття вихідних даних на навчальну і тестову множину за допомогою випадкових вибірок вихідних даних.

Побудова моделей. Ця секція призначена для створення і обробки моделей аналізу даних. Вона надає майстри, які допомагають вам побудувати більшість поширених типів моделей Data Mining без необхідності знатися на відповідних алгоритмах і пов'язаних з ними параметрів, які виконуються на сервері. Також, в цій секції є можливості, що дозволяють користувачеві вибрати конкретний алгоритм і задавати додаткові параметри. Нижче приведен приклад застосування майстра асоціативних правил, меню для знаходження асоціацій в транзакційних даних (рис. 4.21).

Рис. 4.21. Майстер асоціативних правил.

Існують також інші програмні засоби для пошуку асоціативних правил, серед яких слід виділити комерційне та вільно поширюване програмне забезпечення. Зокрема, серед комерційного програмного забезпечення найбільш популярними є:

- Azmy SuperQuery - пошукова система асоціативних правил;
- Clementine, набір від SPSS, що включає аналіз ринкової корзини;
- IBM Intelligent Miner for Data;
- The LPA Data Mining Toolkit - підтримує пошук асоціативних правил в реляційних базах даних;
- Magnum Opus є швидким інструментом пошуку асоціативних правил в даних, підтримується операційними системами Windows, Linux і Solaris;
- Nuggets - це набір, що включає пошук асоціативних правил і інші алгоритми;
- Purple Insight MineSet є набором візуального Data Mining, що включає візуалізатор асоціативних правил.

Серед вільно поширюваного програмного забезпечення слід виділити:

- Apriori - інструмент для знаходження асоціативних правил за допомогою алгоритму Apriori;
- ARtool – інструмент, що містить набір алгоритмів для пошуку асоціативних правил в бінарних базах даних (binary databases);
- DM-II system - інструмент включає алгоритм CBA для виконання класифікації на основі асоціативних правил і деяких інших характеристик;
- FIMI, Frequent Itemset Mining Implementations - є репозиторієм, що включає програмне забезпечення і бази даних.

4.3. Практичний аспект застосування технології асоціативних правил.

Практичні додатки систем інтелектуального аналізу даних на основі технологій асоціативних правил воістину безмежні: виробництво, торгівля, фінанси, медицина, соціологія, наукові дослідження, освіта та ін. Такий інструментарій використовується для задач технічної і медичної діагностики, проектування, управління процесами, контролю якості, прогнозування, оцінки кредитоспроможності, аналізу стану ринків, маркетингових досліджень, роботи з клієнтами, соціологічних опитів, моделювання і вивчення складних систем на основі історії їх еволюції [9, 37, 47, 53, 68, 70, 77].

У великих масивах корпоративних даних часто зберігаються відповіді на багато питань, які цікавлять керівництво і співробітників організацій. Розглянемо деякі з них.

Маркетингові задачі. Одним з прикладів аналізу механізмів стимулювання продажів на основі великих масивів накопичених даних про поведінку споживачів виступають узагальнені асоціативні правила. Ставиться завдання знайти приховані закономірності і типові шаблони поведінки покупців. Для цього вводиться поняття «Купівельна транзакція»: набір товарів, куплених покупцем за один візит до супермаркету. Відмітною властивістю є те, що визначувані асоціативні правила включають елементи, які є предками

елементів, що входять в множину транзакцій. В результаті можливо виявляти асоціації не лише між окремими елементами транзакцій, але і між різними рівнями ієрархії елементів. Якщо продукти харчування розбити, наприклад, на дві групи товарів «Молочні продукти» і «Напої», то ілюстрацією першого випадку буде асоціативне правило «Якщо покупець купив сік, то він, швидше за все, купить кефір», а ілюстрацією другого — «Якщо покупець купив молочні продукти, то він, швидше за все, купить мінеральну воду». Подібні асоціації не так конкретні, як у випадку: «Якщо покупець купив хліб, то з вірогідністю 75% він купить і молоко». Але використання алгоритму пошуку узагальнених асоціативних правил дозволяє значно розширити круг вирішуваних завдань. Цікавою задачею, наприклад, виступає знаходження залежностей між товарами, що продаються деякою фірмою, і її покупцями в наступній постановці: потрібно знайти тих покупців на конкретні товари даної фірми (наприклад, певні товари, які «завалялися» на складі), які до цих пір купували подібні товари інших виробників.

Розглянемо практику комп'ютерного розв’язання типової проблеми, що вирішується за допомогою знаходження асоціативних залежностей. Йдеться про механізми стимулювання продажів, що базуються на знаннях про найбільш типову поведінку покупців при оформленні замовлень. Практична версія системи використовувала 2 бази даних. У кожній є інформація про історію продажів оптової фірми за досить тривалий термін. У першому випадку основною діяльністю фірми є оптовий продаж кондитерських виробів, і аналізована історія продажів складає 1 рік (близько 1,5 млн. записів); у другому випадку це підприємство, що займається оптовим продажем медикаментів, і аналізована історія продажів складає 1,5 років (близько 0,7 млн. записів).

Виявлення дійсно цікавих правил - це одна з головних підзадач при обчисленні асоціативних залежностей. Для того, щоб отримати дійсно цікаві залежності, потрібно розібратися з декількома емпіричними правилами, що претендують на звання аксіом:

1. Зменшення мінімальної підтримки приводить до того, що збільшується кількість потенційно цікавих правил, проте це вимагає істотних обчислювальних ресурсів. Одним з обмежень зменшення порогу minsupport є те, що дуже маленька підтримка правила робить його статистично необґрунтованим.

2. Зменшення порогу достовірності також приводить до збільшення кількості правил. Значення мінімальної достовірності також не має бути дуже маленьким, оскільки цінність правила з достовірністю 5% настільки мала, що це правило і правилом вважати не можна.

3. Правило з дуже великою підтримкою з точки зору статистики є великою цінністю, але з практичної точки зору це, швидше за все, означає те, що або правило всім відомо, або товари, присутні в ньому, є лідерами продажів, звідки слідує їх низька практична цінність.

4. Правило з дуже великою достовірністю практичної цінності в контексті вирішуваного завдання не має, оскільки товари, що входять в наслідок, покупець швидше за все вже купив.

Якщо значення верхньої межі підтримки має дуже велике значення, то в правилах основну частину складатимуть товари - лідери продажів. При такому розкладі не представляється можливим зменшити поріг minsupport до того значення, при якому можуть з'являтися цікаві правила. Причиною тому є просто величезне число правил, і, як наслідок, брак системних ресурсів. Причому отримувані правила відсотків на 95 містять товари - лідери продажів. Приклад таких правил представлений на рисунку 4.22 (зверніть увагу на значення величин, minsupport і maxsupport).

Умова	Наслідок	Підтримка (%)	Достовірність (%)
ПАСТИЛА ВАНІЛЬНА і З-Р ВАНІЛЬНИЙ і ВФ ГУЛІВЕР	ВФ АРТЕК-СУПЕР	1,11	52,27
ПЕЧ.ВІВСЯНЕ і З-Р КРЕМ-БРЮЛЕ і ВФ ШОК-ГОРІХОВИЙ і	ВФ АРТЕК-СУПЕР і З-Р ВАНІЛЬНИЙ	1,15	59,50

З-Р ВЕРШКОВИЙ З-Р ФРУКТОВО-ЯГІДНИЙ	З-Р КРЕМ-БРЮЛЕ	1,11	68,32
ПАСТИЛА ВАНІЛЬНА і З-Р КРЕМ-БРЮЛЕ і ПЕЧ. З РОДЗИНКАМИ	ПЕЧ.ВІВСЯНЕ і З-Р ВАНІЛЬНИЙ	1,12	55,12
...	...	...	...

Рис. 4.22. Приклад асоціативних правил (БД: Кондитерські вироби; minsupport: 1.1; maxsupport: 100; minconfidence: 50; maxconfidence: 95).

Варіюючи верхньою і нижньою межами підтримки, можна позбавитися від очевидних і нецікавих закономірностей. Як наслідок, правила, що генеруються алгоритмом, набирають наближеного до реальності вигляду (рис. 4.23).

Умова	**Наслідок**	**Підтримка (%)**	**Достовірність (%)**
ДОЛЬКА МУЛЬТИФРУКТ НЕКТАР	ДОЛЬКА ТОМАТ СІК	0,93	77,33
КФ ВІЗІТ	КФ ВЕЧІРНІ	0,91	45,24
З-Р РОМОВИЙ В ШОК.	З-Р КАВОВИЙ В ШОК.	1,39	63,97
...	...	...	...

Рис 4.23. Приклад асоціативних правил (БД: Кондитерські вироби; minsupport: 0,9; maxsupport: 10; minconfidence: 40; maxconfidence: 95)

В деяких випадках значення порогу minsupport має бути дійсне маленьким. Прикладом можуть бути правила, отримані на основі даних бази даних «Медикаменти». Річ у тому, що в деяких областях торгівлі товарів настільки багато, що кожен з них не володіє ринком навіть на декілька відсотків. Такою областю є торгівля фармацевтичною продукцією. Наприклад, навіть в маленькій аптеці кількість найменувань товарів складає біля 2-х тисяч. В цьому випадку з'являється ще гостріша необхідність в зменшенні мінімальної підтримки, яку, у свою чергу, просто не удається зменшити, не зменшуючи верхній поріг (рис. 4.24). На рисунку 4.24 представлені приклади правил з великим значенням порогу maxsupport.

Умова	Наслідок	Підтримка (%)	Достовірність (%)
Андипал # 10 і Еналаприл тб. 0,01 № 20	Аскорбінова к-та 5% 1мл #10	1,01	64,31
Ортофен 2,5% 3,0 № 10 і Екстракт валер'яни 0.02 # 10	Аскорбінова к-та 5% 1мл #10 і Цианокобаламін (віт.В12)500мкг 1мл #10	1,02	42,63
Діоксидин 1% 5.0 # 10 і Ортофен 2,5% 3,0 № 10	Аскорбінова к-та 5% 1мл #10 і Екстракт валер'яни 0.02 # 10	1,09	48,22
Аскорбінова к-та 5% 1мл #10 і Цитрамон П тб.# 6	Рибоксин 0,2 тб.#50 і Валідол 0,05 капс.#20	1,05	40,96
...	...	...	...

Рис 4.24 Приклад асоціативних правил (БД: Медикаменти; minsupport: 0,9; maxsupport: 100; minconfidence: 40; maxconfidence: 95).

А ось що виходить, якщо зменшити значення максимально доступної підтримки і порогу minsupport (рис. 4. 25).

Умова	Наслідок	Підтримка (%)	Достовірність (%)
Ібупрофен 0,2 тб.#50 і Валідол тб.0.06 # 10	Анальгін 0.5 # 10	0,10	51,89
Валідол тб.0.06 # 10 і Парацетамол 0.5 # 10 і Ацетилсалицилова к-та 0,5 #10	Бромгексин 0,008 тб.#10	0,11	72,84
Аеровіт # 30 і Гептавіт # 20	Декамевіт тб.#20	0,10	52,38
Еуфіллін 0,15 тб.#30 і Бромгексин 0,008 тб.#10	Парацетамол 0.5 # 10	0,13	59,83
...	...	...	...

Рис 4.25. Приклад асоціативних правил (БД: Медикаменти; minsupport: 0,1; maxsupport: 5; minconfidence: 40; maxconfidence: 95).

Описані вище механізми є однією із складових частин інформаційної системи виписки замовлень. Обчислення правил при цьому проходить у фоновому режимі, тобто користувач цієї системи (менеджер) цього не бачить. Проте при виписці замовлень або ж по запиту йому будуть представлені

обчислені правила, що допоможе менеджерові бути підготовленішим до питань клієнтів. Обчислені правила можна використовувати як для оперативної підказки, так і для аналізу. Але в обох випадках асоціативні правила потрібно грамотно представити користувачеві. Для цього можна використовувати різні відображення, головна задача яких - бути максимально зрозумілими і легко засвоюваними.

В інформаційній системі представлено чотири варіанти відображення асоціативних правил: у вигляді звичайної таблиці, у вигляді форматованого тексту, у вигляді дерева, у вигляді перехресної таблиці. В кожного відображення є як свої переваги, так і недоліки, і вибір конкретного залежить як від тих цілей, які користувач хотів би досягти, так і від особливостей особистого сприйняття.

Однією з найпростіших для сприйняття візуалізацій асоціативних правил є таблиця (рис. 4.22 – 4.25). Її перевагами є:

- легко зрозуміти, де умова, а де наслідок (у різних стовпцях);
- можливість експорту в більшість форматів;
- можливість сортування по умові, по слідству, по підтримці, по достовірності;
- можливість фільтрації по умові, по слідству, по підтримці, по достовірності.

Недоліки: дуже велика кількість правил приводить до генерації громіздкого документа і, як наслідок, до складності його розуміння.

Найприроднішою конструкцією для сприйняття правил є конструкція вигляду: ЯКЩО «Умова» ТО «Наслідок». Такий варіант відображення правил – найбільш доступна форма представлення (рис. 4.26).

...
Правило N 6
ЯКЩО Метронідазол 0,25 тб.#10
ТО Ібупрофен 0,2 тб.#50
Підтримка = 1,20%
Достовірність = 44,62%

Правило N 7
ЯКЩО Піридоксину г/х (витий.В6) 2мг тб.#50
ТО Магнію сульфат 25% 5мл #10
Підтримка = 0,60%
Достовірність = 42,30%
...

Рис. 4.26. Асоціативні правила у вигляді форматованого тексту.

Перевагами цього варіанту відображення є: природний запис правил, можливість експорту в .rtf файл, можливість сортування по умові, по слідству, по підтримці та по достовірності, можливість фільтрації по умові, по слідству, по підтримці та по достовірності. Недоліки: дуже велика кількість правил приводить до генерації громіздкого документа, і як наслідок складність в розумінні.

Один з варіантів відображення асоціативних правил – це дерево правил. На перший погляд використання такого відображення невиправдане з точки зору розуміння сенсу правил, але при частому використанні ця візуалізація стає просто незамінною. Правило читається таким чином:

1. Вираз, який буде умовою правила збирається починаючи з поточного (виділеного) вузла.

2. Далі, рухаючись до кореневого вузла, до виразу додається «І» плюс «значення батьківського вузла».

3. Процедура (2) повторюється рекурсивно до тих пір, поки не досягне батьківського вузла.

4. Наслідки знайденої умови (якщо такі є) знаходяться праворуч від дерева в таблиці.

Наприклад, правила для виділеного вузла виглядатимуть так (рис. 4.27):

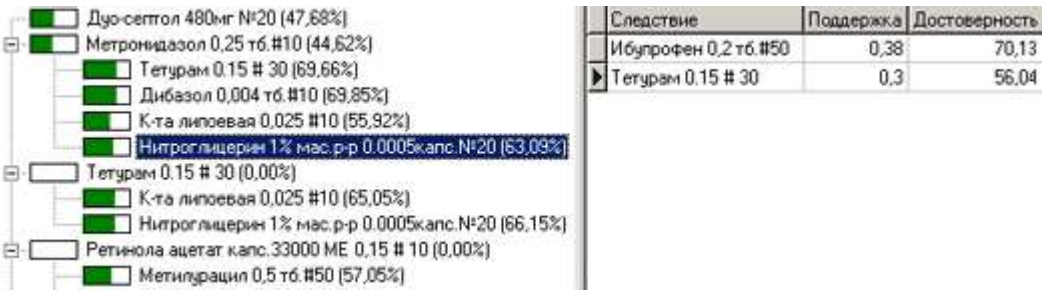


Рис. 4.27. Асоціативні правила у вигляді дерева (БД: Медикаменти; minsupport: 0,1; maxsupport: 3; minconfidence: 40; maxconfidence: 95).

Перевагами цього засобу є: компактний вигляд, групування правил по умові, можливість відображення значень підтримки і достовірності правила в графічному вигляді. Недоліки: незвичайне відображення правил, і, як наслідок, складність розуміння правил на перших порах.

Ще одним варіантом візуалізації асоціативних правил є перехресна таблиця. У цій таблиці по осі X перераховані всі правила, а по осі Y - всі товари, що увійшли до правил (рис. 4.28). Правило читається таким чином:

1. Вибирається який-небудь стовпець. Це буде одне правило.
2. У умову входять товари, позначені знаком «+».
3. У наслідок входять товари, позначені знаком «=».
4. Підтримка і достовірність правила вказані в 2-му і 3-му рядках відповідно.

№	4	5	6	7	8	9	10	11	12
Поддержка (%)	1,20	1,20	0,98	0,83	0,55	0,72	0,55	0,75	0,59
Достоверность (%)	44,62	40,50	40,44	52,35	69,66	51,36	62,22	42,43	56,59
Викасол тб.#10				+					
Викалин # 10				=					
Дуо-септол 120мг №20									
Дуо-септол 480мг №20									
Ибупрофен 0,2 тб.#50	=	+			=		+	=	
К-та липоевая 0,025 #10								+	
Магния сульфат 25% 5мл #10									
Метронидазол 0,25 тб.#10	+	=			+		=		
Нитроглицерин 1% мас.р-р 0.0005капс.№20									
Папаверина г/хл 0,04 тб.#10									
Пиридоксина г/х (вит.В6) 2мг тб.#50						+			
Проходол детский суспензия 100 мл									=
Проходол таб.№ 12									+
Ретинола ацетат капс.33000 МЕ 0,15 # 10						=			
Тетурам 0.15 # 30					+		+		
Тисасен драже 10 мг # 30			+						
Тисацид тб. 500 мг # 30			=						

Рис. 4.28. Асоціативні правила у вигляді перехресної таблиці (БД: Медикаменти; minsupport: 0,1; maxsupport: 3; minconfidence: 40).

Перевагами перехресних таблиць є: можливість побачити всі товари, присутні у всіх правилах, можливість сортування по умові, по слідству, по підтримці та по достовірності, можливість фільтрації по умові, по слідству, по підтримці та по достовірності. Недоліки: дуже велика кількість товарів, присутніх в правилах, викликає деяку незручність при аналізі закономірностей.

Слід зазначити, що застосованість алгоритму цим не обмежується. За допомогою даного алгоритму можна знаходити закономірності між зв'язаними подіями, тобто якщо вихідні дані представлені у вигляді подій, то на цих даних можна побудувати асоціативні закономірності. Одним з таких прикладів може служити знаходження залежностей вигляду: ЯКЩО «РЕЗУЛЬТАТИ АНАЛІЗІВ» ТО «ЗАХВОРЮВАННЯ», де «РЕЗУЛЬТАТИ АНАЛІЗІВ» – це результати аналізів пацієнта, а «ЗАХВОРЮВАННЯ» – це список захворювань, якими, можливо, хворіє даний пацієнт. Список же потенційно можливих областей дуже великий.

Задачі митниці. Технологія асоціативних правил може успішно застосовуватися для виявлення прихованих тенденцій в зовнішньоторговельній діяльності. Одна з основних проблем, що стоїть перед митними органами, полягає у виявленні навмисного спотворення вантажних митних декларацій. Через обмежені ресурси повна перевірка всіх переміщуваних через кордон вантажів неможлива. Проте митниця збирає детальні бази даних по вантажних митних деклараціях. Аналіз цих даних може бути використаний для виявлення тенденцій в зовнішній торгівлі України по групах товарів, найбільш схильних до фальсифікації при проходженні митниці — «товарів ризику». Маючи дані про такі товари, митні пости могли б ретельніше перевіряти проходження відповідних вантажів і зменшити втрати від фальсифікації митних документів. Однією з особливостей задачі стала відсутність «тренувального» набору даних - даних, для яких було б апріорі відомо, які з них є спробою фальсифікації вантажної митної декларації, а які є сумлінно задекларованими товарами. Це істотно обмежувало круг алгоритмів, які можна було використовувати: наприклад, популярні методи типа дерева рішень, нейроні мережі і тому

подібне вимагають попереднього навчання на тренувальному наборі даних. У нашому розпорядженні залишалися лише алгоритми асоціативних правил.

Як правило, при «прикритті» одного товару іншим в рамках одного вантажу (і однієї митній декларації) дійсно перевозяться обоє товарів, проте доля «дорогого» занижується. Цей факт і може бути використаний для виявлення подібних пар. При відборі потенційних пар «товар риску» - «товар прикриття» використовувалися наступні критерії:

1. $P(A/B) = \frac{P(AB)}{P(B)}$ або $P(B/A) \approx 1$ - умовна ймовірність ввозу товару A, коли по тій же декларації ввозиться товар B, достатньо велика (аналогічний критерій застосовується а алгоритмі асоціативних правил системи Oracle Server 9i ODM);
2. $S(A) > S(B)$ - мито на «товар риску» більше ніж на «товар прикриття»;
3. $netto(B) > 0$ - порівняння зі статистикою ЄС показує завищення об’єму імпорту для «товарів прикриття»;
4. $netto(A) < 0$ - порівняння зі статистикою ЄС показує заниження об’єму імпорту для «товарів риску».

Перший критерій основний і означає, що один з товарів найімовірніше супроводить іншому. Вибір умовної вірогідності, замість, наприклад, коефіцієнта кореляцій, обумовлюється їх більшою чутливістю. Коефіцієнт кореляції близький до одиниці лише в разі, якщо обоє товарів весь час ввозяться одночасно. Критерій накладає набагато слабкішу умову: лише один з товарів постійно супроводить іншому, оскільки один з товарів може ввозитися у великих об'ємах без жодного супроводу. Використаний критерій відомий в літературі як алгоритм асоційованих правил і, зокрема, реалізований в Oracle Data Mining 9i. Втім, висока кореляція одного з товарів з іншим ще не означає, що товар обов'язково прикривається іншим: безліч людей щодня купують одночасно хліб і молоко без жодного злого наміру. І при імпорті товарів існують випадки природної кореляції між товарами. Аби очистити відібрані пари від таких випадків, були накладені додаткові умови: прикриття має бути

економічно вигідно, а порівняльний аналіз статистичних даних повинен підтверджувати факт прикриття.

Аналіз наданих митницею даних виявив значну кількість пар, що задовольняють вибраним критеріям. Безумовно, не всі вони є парами «товар риску» - «товар прикриття». Ефективність реалізованого алгоритму може бути підтверджена лише в ході додаткових перевірок на митних постах. Проте слід зазначити, що число подібних пар істотно менше, ніж загальне число товарних груп, і їх список сповна може бути використаний як рекомендація по ретельнішому догляду певних вантажів. Як приклад приведемо одну пару товарів: шини для легкових автомобілів і протекторні заготовки для їх відновлення. Впродовж всього 2008 року ймовірність ввезення шин разом із заготовками дуже висока — в середньому 95% за рік. Випадків ввезення лише заготовок практично не було. При цьому коефіцієнт кореляції не настільки великий, оскільки чималий об'єм імпорту шин не супроводиться заготовками. Сам по собі факт кореляції між цими групами товарів досить природний, проте ставка митного збору на заготовки була в 5 разів нижче, ніж для шин. Більш того, порівняльний аналіз даних України і ЄС показав, що імпорт заготовок згідно з українськими даними майже в 200 разів вище, ніж за даними ЄС, а імпорт шин нижче в 3,5 рази, якщо порівнювати об'єми імпорту по вазі. При цьому сумарна вага імпорту по цих двох групах збігається за даними України і ЄС з точністю до 20%. Схожа картина спостерігається і у вартісному вираженні. Вартість ввезених в Україну заготовок в 30 разів вища, ніж вивезених з країн ЄС, тоді як шин, якщо судити за вартістю, що декларується, ввезено в 2,7 разу менше вивезеної кількості. Таким чином., судячи по приведеним даним, з великою вірогідністю протекторні заготовки використовувалися рядом імпортерів як прикриття для шин, що ввозилися. Втрати держави на митних зборах склали імовірно близько 7 млн. дол.

Відзначимо, що аналіз був проведений на повному об'ємі вантажних митних декларацій за 2008 рік, що складає більше 2 млн. декларацій із загальним числом товарів порядка 5 млн. Ясно, що аналіз такої кількості даних

не може бути виконаний ні вручну, ні за допомогою ряду інших технологій підтримки рішень. І хоча, безумовно, неможливо повністю замінити аналітика автоматизованою системою, вживання методів пошуку знань дозволяє відсіяти величезну кількість даних, що не представляють інтересу і скоротити об'єм аналізованої інформації до рівня адекватного людському сприйняттю.

Задачі медицини. Сучасний рівень розвитку медицини характеризується тим, що при здійсненні практично будь-якого різновиду лікувального процесу збирається велика кількість супутньої інформації: результати аналізів та обстежень, протоколи оперативних втручань та оглядів. Особливостями цієї інформації є:

- Велика кількість атрибутів (у тому числі числових).
- Зашумленість. Наявність помилкових значень, викликаних ручним введенням даних або погрішностями апаратури.
- Часткова заповненість. Наявність незаповнених полів в дослідженнях або відсутність цілих досліджень у деяких пацієнтів.
- Погана структурованість. Вихідна інформація зберігається в різних типах СУБД, файлах, на паперових носіях і, як правило, складається з наборів моніторингів та аналізів різної структури.

Таким чином, актуальним завданням є розробка ефективних методів і алгоритмів виявлення залежностей у вигляді, доступному інтерпретації людині, між значеннями атрибутів даних, що володіють перерахованими властивостями, та реалізація вказаних методів і алгоритмів в програмних системах, що забезпечують всі фази обробки даних, включаючи: автоматизацію збору даних, первинну обробку і нормалізацію, інтерактивну взаємодію з експертом, візуалізацію результатів. Слід зазначити, що, весь набір інформації, що супроводжує лікувальний процес, в більшості випадків можна розділити на 2 групи: вхідна інформація (результати обстежень та моніторингів, отриманих до і під час дії на пацієнта, характеристики дії на пацієнта), вихідна інформація (результати обстежень у відновний період). Кожна з груп зазвичай містить значне число атрибутів.

В якості основного математичного апарату виявлення залежностей були використані числові асоціативні правила. Через наявність в даних прикладної області двох груп «вхідних» і «вихідних» атрибутів була вибрана методика використання асоціативних правил з обмеженнями на місце розташування атрибуту в правилі. Методика передбачає можливість явно задати для будь-якого атрибуту обмеження на його положення в лівій або правій частині асоціативного правила. Дане обмеження використовується в алгоритмі Apriori для оптимізації фази генерації наборів значень атрибутів, що часто зустрічаються. Окрім цього в системі реалізована можливість явно задавати на множині атрибутів відношення приналежності до груп залежних атрибутів. Групи можуть вибиратися експертами-медиками, ґрунтуючись на теоретичних знаннях про взаємозв'язки між деякими аналізами та дослідженнями. Планується реалізувати з використанням методів кореляційного аналізу можливість автоматизованого виявлення залежних атрибутів в групах атрибутів лівої і правої частин і виключення з результуючого набору правил, що містять їх комбінації.

У медицині, як ні в якій іншої прикладної області, велика роль теоретичних знань, накопичених за час її розвитку. В нашому випадку прикладом таких медичних знань разом з апріорі відомими взаємозв'язками між деякими аналізами є діапазони значень норм різних показників аналізів. Для інтеграції цих знань в процес виявлення асоціативних правил в системі існує спеціальний інструментарій, що дозволяє медичному експертові задавати лінгвістичні змінні декількох різновидів на множині значень атрибутів аналізованих даних. Результуючі асоціативні правила описуватимуть взаємозв'язки між заданими лінгвістичними змінними. Системою підтримуються лінгвістичні змінні у вигляді чітких і нечітких інтервалів. Останні знаходять в медицині широке вживання через нечіткість багатьох її понять.

Однією з ключових можливостей системи є автоматична дискретизація атрибутів за допомогою вдосконаленого алгоритму дискретизації RUDE, що

дозволяє знаходити семантично обґрунтоване розбиття на інтервали значень атрибутів, лінгвістичні змінні по яких не задані.

В якості базового алгоритму використовується алгоритм, що реалізовує метод Apriori пошуку асоціативних правил з використанням дерев. Алгоритм був модифікований для ефективної роботи з числовими атрибутами у фазі генерації наборів, що часто зустрічалися. Для валідації результатів роботи застосованих алгоритмів і методів використовується ідея синтаксичної близькості правил, заснованої на відстані між правилами:

$$D(r_1, r_2) = \delta_1 \mid (X_1 Y_1) \Theta (X_2 Y_2) \mid + \delta_2 \mid X_1 \Theta X_2 \mid + \delta_3 \mid Y_1 \Theta Y_2 \mid$$

Для порівняння двох наборів правил r_1 та r_2 по одному з критеріїв цікавості пропонується порівнювати їх пересічення $r_1 \cap r_2$ з кожним з множини, знаходити сумарне значення критерію цікавості для підмножин правил r_1 та r_2, що входять в пересічення, окремо експертним способом оцінювати цікавість і корисність правил, що не увійшли до пересічення.

Система реалізована у вигляді віконного застосування на платформі .NET. Зберігання моделей і взаємодія між ядром і призначеним для користувача інтерфейсом здійснюється з використанням стандарту PMML 3.0, розширеного для нечіткої дискретизації і обмежень на місце розташування атрибутів. Це дозволяє в перспективі інтегрувати дану систему з іншими системами інтелектуального аналізу даних.

Інтернет-торгівля. Для ефективного управління бізнесом у сфері електронної комерції великого поширення набувають методи бізнес-аналітики. У сферу їх застосування входять задачі по прогнозуванню об'ємів продажів, управлінню кількістю товарних запасів, визначенню оптимальних торгівельних націнок, виявленню типових паттернів купівельної поведінки, оптимізації навігації по сайту, поліпшенню рубрикації і тому подібне

Розглянемо вживання технологи асоціативних правил для виявлення закономірностей в даних книжкового інтернет-магазину. Всі використовувані для аналізу дані охоплюють період в півтора роки функціонування магазина і зберігаються в централізованій базі даних під управлінням СУБД Microsoft SQL

Server 2005. В якості технологічної платформи для аналізу даних був використаний Microsoft Analysis Services 2005.

Методом асоціативних правил були проаналізовані товари і товарні групи, що спільно входять в одну транзакцію (одне замовлення). Асоціативний аналіз спільного входження книжкових найменувань в одне замовлення виявив:

1. Правило входження книг одного і того ж автора в одне замовлення. Таким чином, для збільшення об'єму продажів, при замовленні відвідувачем сайту якої-небудь книги слід пропонувати йому книги того ж автора, що є в наявності (рис. 4.29).

Рис. 4.29. Відвідувачам сайту книг слід пропонувати книги того ж автора.

2. Правило спільних покупок дитячих книг в одному замовленні. Цей факт привів до виділення дитячої літератури в окрему групу, що збільшило продажі книг цієї групи на 7,5% (рис. 4.30).

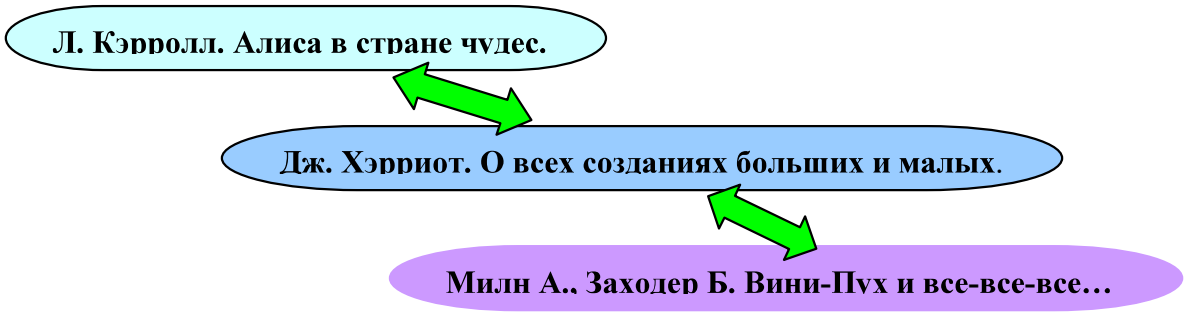


Рис.4.30. Правило спільних покупок дитячих книг в одному замовленні.

3. Правило спільних покупок езотеричної літератури. Цей факт привів до рекомендації про виділення книг про «правильне харчування», «особисте вдосконаленні» і так далі в окрему групу.

Асоціативний аналіз правил спільного входження товарних груп в одне замовлення не виявив значимих і достовірних правил по входженню товарів різних груп в одне замовлення, що говорить про адекватне існуюче розбитті книг на групи.

4. Послідовності категорій книг, що замовляються клієнтами.

Аналіз послідовностей книг, що замовляються клієнтами, показав, що якщо в замовлення клієнта входять книги з груп: зарубіжна поезія, комп'ютери і інтернет, біографії, мемуари та публіцистика, медицина, економіка, бізнес, та менеджмент, то в подальших замовленнях того ж клієнта зустрічаються книги з тієї ж групи.

Якщо клієнт замовив детективи, то в подальших замовленнях він замовляє або знову детективи, або зарубіжну прозу, або дитячу літературу. При замовленні клієнтом дитячої літератури в наступних замовленнях він, швидше за все, знову вибере книги з цієї групи, або купить зарубіжну прозу. Висновок: групи «зарубіжна проза», «дитяча література» і «детективи» повинні розташовуватися з точки зору навігації на сайті як можна ближче один до одного, що означає можливість швидкого переходу між цими тематичними групами.

Перспективні напрямки. Існує широкий спектр задач, в яких потрібно виявляти асоціації із збереженням об’єктно-ознакового опису даних. Це задачі біоінформатики по виявленню груп генів, що володіють загальними властивостями, пошук груп відвідувачів з схожими інтересами для рекомендаційних систем, виявлення інтернет-співтовариств, наукових співтовариств, задачі аналізу соціальних мереж, побудова автоматичних каталогів і рубрикаторів в інформаційних системах, пошук документів-дублікатів.

Задача пошуку асоціацій останніми роками набула велику популярність у зв'язку із зростаючою потребою в аналізі генетичних даних. Основне завдання такого аналізу — виявлення груп генів, що проявляють схожу поведінку лише за певних умов. Надалі фахівець генетик на підставі проведеного аналізу висуває гіпотезу про те, чи є дана група генів причиною досліджуваної хвороби. Окрім даних генної експресії, як додатки в біології можна привести проекти, які аналізували активність лікарських препаратів. Розглядаються і інші масиви генетичних даних, що в основному відносяться до ракових захворювань, таким як лімфома. Наприклад, для таких задач широке застосування має система Coron.

Система аналізу даних Coron призначена для пошуку частої множини ознак і асоціативних правил. Програма володіє непоганим графічним інтерфейсом, власним форматом даних, можливістю роботи з базами даних. Для пошуку частої множини ознак використовуються найбільш ефективні алгоритми співтовариства FIM. Пошук асоціативних правил також використовує ефективні алгоритми, що спираються на досягнення ФАП і виявилися корисними для компактного відображення правил і побудови їх базисів. Ще одним достоїнством продукту є вільний доступ і кросплатформенність.

Останніми роками поновився інтерес до міждисциплінарних досліджень в області аналізу соціальних мереж, в яких задіяні математична соціологія і інформатика, що спираються на апарат теорії графів. Зусилля в цьому напрямі в значній мірі підтримуються завдяки новим обчислювальним можливостям і доступності електронних даних для деяких соціальних систем: співтовариств вчених, людей, ведучих особисті електронні щоденники (weblogers), покупців інтернет-магазинів, мереж друзів, сайтів знайомств і так далі. Зокрема, в центрі багатьох поточних досліджень знаходяться мережі знань, тобто мережі взаємодій, в яких агенти виробляють знання або обмінюються ними. До числа досліджень входить виявлення співтовариств, що розглядається як знаходження агентів, які володіють множиною загальних ознак. Аналіз соціальних мереж

спеціалізується на методах виявлення, опису і правдоподібної організації різних видів соціальних співтовариств. Для аналізу соціальних аспектів співтовариств основний інтерес представляють лідери, периферійні члени, міжгрупова і внутрішньогрупова взаємодія.

Технологія асоціативних правил успішно застосовувалися для аналізу епістемічних співтовариств (тобто агентів, що мають справу з однаковими темами, наприклад, наукові співтовариства або користувачі блогів) або філіальних мереж (актори належать одній і тій же організації). Успіх цього підходу обумовлений наявністю таксономій, що виявляється корисним при ієрархічному описі груп акторів в термінах схожості інтересів.

Методи пошуку асоціацій можуть бути використані для так званої коллаборативної фільтрації (collaborative filtering) при виявленні груп покупців зі схожими перевагами у вигляді деякої підмножини товарів (задача цільового маркетингу). Схожа ситуація має місце в рекомендаційних системах, де асоціативні правила надають інформацію про схожі інтереси груп відвідувачів. Необхідно відзначити, що рекомендаційні системи і цільовий маркетинг - важливі застосування в області електронної комерції. У таких проектах основною метою є виявлення груп покупців, ведучих себе схожим чином, аби передбачити їх інтереси і запропонувати адекватні рекомендації. Іншим прикладом є ринок інтернет-реклами, для якого актуальний пошук асоціацій, що представляють окремі ринки, тобто множину покупців і придбаних ними рекламних словосполук.

Існують і менш поширені застосування теорії асоціативних правил, що спираються, наприклад, на дані про голосування. В цьому випадку необхідно виявляти підмножини рядків виборців, що дотримуються схожих політичних поглядів і що проявляють схожу електоральну поведінку на підмножині даних ознак.

4.4. Дерева рішень - загальні принципи технології.

Стрімкий розвиток інформаційних технологій, зокрема, прогрес в методах збору, зберігання і обробки даних дозволив багатьом організаціям збирати величезні масиви даних, які необхідно аналізувати. Об'єми цих даних настільки великі, що можливостей експертів вже не вистачає, що породило попит на методи автоматичного дослідження (аналізу) даних, який з кожним роком постійно збільшується. Дерева рішень (decision trees) – один з таких методів автоматичного аналізу даних. Ця технологія є однією з найбільш популярних методів вирішення задач класифікації і прогнозування. Інколи цей метод інтелектуального аналізу даних також називають деревами рішающих правил або деревами класифікації і регресії. Як видно з останньої назви, за допомогою даного методу вирішуються задачі класифікації і прогнозування. Якщо залежна, тобто цільова змінна набуває дискретних значень, при допомозі методу дерева рішень вирішується задача класифікації. Якщо ж залежна змінна набуває безперервних значень, то дерево рішень встановлює залежність цієї змінної від незалежних змінних, тобто вирішує задачу чисельного прогнозування.

Перші ідеї створення дерев рішень запропоновані в роботах Ховленда (Hoveland) і Ханта (Hunt) в кінці 50-х років XX століття. Проте, основополагаючою роботою, що дала імпульс для розвитку цього напряму, з'явилася книга Ханта (Hunt E.B.), Меріна (Marin J.) і Стоуна (Stone P.J) «Experiments in Induction», що побачила світло в 1966 р. [9, 37, 47]

Означення. Дерева рішень – це технологія представлення правил в ієрархічній, послідовній структурі, де кожному об'єкту відповідає єдиний вузол, що дає рішення. Під правилом розуміється логічна конструкція, представлена у вигляді «Якщо ... то ...» (if-then).

Для ухвалення рішення, до якого класу віднести деякий об'єкт або ситуацію, потрібно відповісти на питання, що стоять у вузлах цього дерева, починаючи з його кореня. Питання мають вигляд «значення параметра А

більше x?». Якщо відповідь позитивна, здійснюється перехід до правого вузла наступного рівня, якщо негативна - то до лівого вузла; потім знову слідує питання, пов'язане з відповідним вузлом (рис. 4.31).

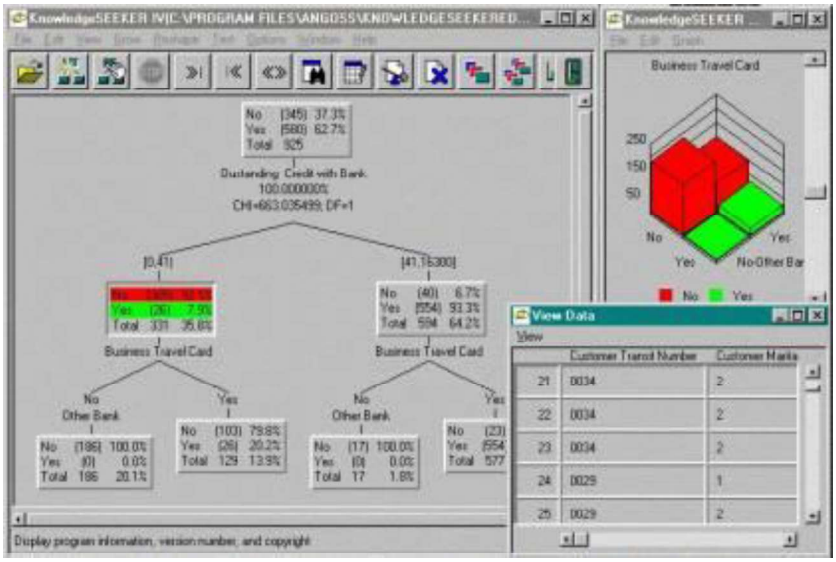


Рис 4.31. Система KnowledgeSeeker обробляє банківську інформацію.

Більшість систем інтелектуального аналізу даних використовують цей метод для пошуку закономірностей. Найвідомішими є See5/C5.0 (RuleQuest, Австралія), Clementine (Integral Solutions, Великобританія), SIPINA (University of Lyon, Франція), IDIS (Information Discovery, США), KnowledgeSeeker (ANGOSS, Канада).

Сфера застосування дерев рішень в даний час широка, але всі задачі, що вирішуються цією технологією можуть бути об'єднані в наступні три класи:

1. Опис даних: Дерева рішень дозволяють зберігати інформацію про дані в компактній формі. Замість самих даних ми можемо зберігати дерево рішень, яке містить точний опис об'єктів.

2. Класифікація: Дерева рішень відмінно справляються із задачами класифікації, тобто віднесення об'єктів до одного із заздалегідь відомих класів.

3. Регресія: Якщо цільова змінна має безперервні значення, дерева рішень дозволяють встановити залежність цільової змінної від незалежних (вхідних) змінних.

Дерева рішень є прекрасним інструментом в системах підтримки прийняття рішень та інтелектуального аналізу даних. В склад багатьох пакетів, призначених для інтелектуального аналізу даних, вже включені методи побудови дерев рішень. У областях, де висока ціна помилки, вони служать відмінною підмогою аналітика або керівника. Дерева рішень успішно застосовуються для вирішення практичних задач в наступних областях:

- *Банківська справа*. Оцінка кредитоспроможності клієнтів банку при видачі кредитів.
- *Промисловість*. Контроль за якістю продукції (виявлення дефектів), випробування без руйнувань (наприклад перевірка якості зварки) і так далі**.**
- *Медицина*. Діагностика різних захворювань.
- *Молекулярна біологія*. Аналіз будови амінокислот.

Це далеко не повний список областей де можна використовувати дерева рішень. Не досліджено ще багато потенційних сфер застосування [9, 37, 47].

Розглянемо сутність цієї технології на прикладі дерева рішень, завдання якого - відповісти на питання: «Чи грати в гольф?» Аби вирішити задачу, тобто прийняти рішення, чи грати в гольф, слід віднести поточну ситуацію до одного з відомих класів (в даному випадку – «грати» або «не грати»). Для цього потрібно відповісти на низку запитань, які знаходяться у вузлах цього дерева, починаючи з його кореня (рис. 4.32).

Перший вузол нашого дерева «Сонячно?» є вузлом перевірки, тобто умовою. При позитивній відповіді на питання здійснюється перехід до лівої частини дерева, званою лівою гілкою, при негативному - до правої частини дерева. Таким чином, внутрішній вузол дерева є вузлом перевірки певної умови. Далі йде наступне питання і так далі, поки не буде досягнутий кінцевий вузол дерева, що є вузлом рішення. Для нашого дерева існує два типи кінцевого вузла: «грати» і «не грати» в гольф. В результаті проходження від кореня

дерева (інколи званого кореневою вершиною) до його вершини вирішується задача класифікації, тобто вибирається один з класів – «грати» і «не грати» в гольф.

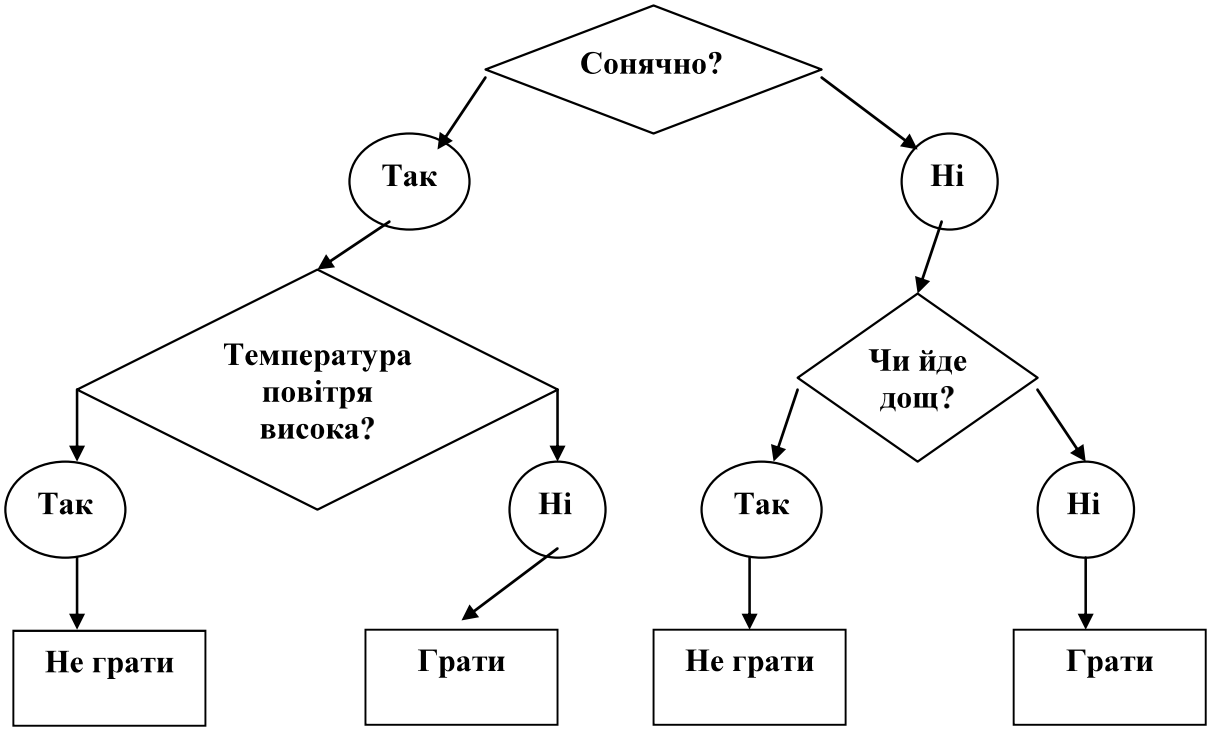


Рис. 4.32. Дерево рішень задачі «Чи грати в гольф?».

Метою побудови дерева рішення в нашому випадку є визначення значення категоріальної залежної змінної. Отже, для нашої задачі основними елементами дерева рішень є: корінь дерева – «Сонячно?», внутрішній вузол дерева або вузол перевірки – «Температура повітря висока?», «Чи йде дощ?», лист, кінцевий вузол дерева, вузол рішення або вершина: «Грати», «Не грати», гілка дерева (випадки відповіді) – «Так», «Ні».

У розглянутому прикладі вирішується задача бінарної класифікації, тобто створюється дихотомічна класифікаційна модель. Приклад демонструє роботу так званих бінарних дерев. У вузлах бінарних дерев галуження може вестися лише в двох напрямах, тобто існує можливість лише двох відповідей на поставлене питання («так і ні»). Такі дерева є найпростішими, окремим

випадком дерев рішень. В інших випадках, відповідей і, відповідно, гілок дерева, що виходять з його внутрішнього вузла, може бути більше двох.

Розглянемо складніший приклад. База даних, на основі якої повинне здійснюватися прогнозування, містить наступні ретроспективні дані про клієнтів банку, що є її атрибутами: вік, наявність нерухомості, освіта, середньомісячний дохід, чи повернув клієнт вчасно кредит. Завдання полягає в тому, аби на підставі перерахованих вище даних (окрім останнього атрибуту) визначити, чи варто видавати кредит новому клієнтові. Така задача, як вже відомо, вирішується в два етапи: побудова класифікаційної моделі і її використання.

На етапі побудови моделі, власне, і будується дерево класифікації або створюється набір деяких правил. На етапі використання моделі побудоване дерево, або дорога від його кореня до однієї з вершин, той, що є набором правил для конкретного клієнта, використовується для відповіді на поставлене питання чи «Видавати кредит?» Правилом є логічна конструкція, представлена у вигляді «якщо : то :».

На рис. 4.33. наведений приклад дерева класифікації, за допомогою якого вирішується задача «Видавати кредит клієнтові?». Вона є типовою задачею класифікації, і за допомогою дерев рішень отримують досить хороші варіанти її рішення.

Як бачимо, внутрішні вузли дерева (вік, наявність нерухомості, дохід і освіта) є атрибутами описаної вище бази даних. Ці атрибути називають такими, що прогнозують, або атрибутами розщеплювання (splitting attribute). Кінцеві вузли дерева, або листи, іменуються мітками класу, що є значеннями залежної категоріальної змінної «видавати» або «не видавати» кредит. Кожна гілка дерева, що йде від внутрішнього вузла, відмічена предикатом розщеплювання. Останній може відноситися лише до одного атрибуту розщеплювання даного вузла. Характерна особливість предикатів розщеплювання: кожен запис використовує унікальну дорогу від кореня дерева лише до одного вузла-рішення. Об'єднана інформація про атрибути розщеплювання і предикати

розщеплювання у вузлі називається критерієм розщеплювання (splitting criterion).

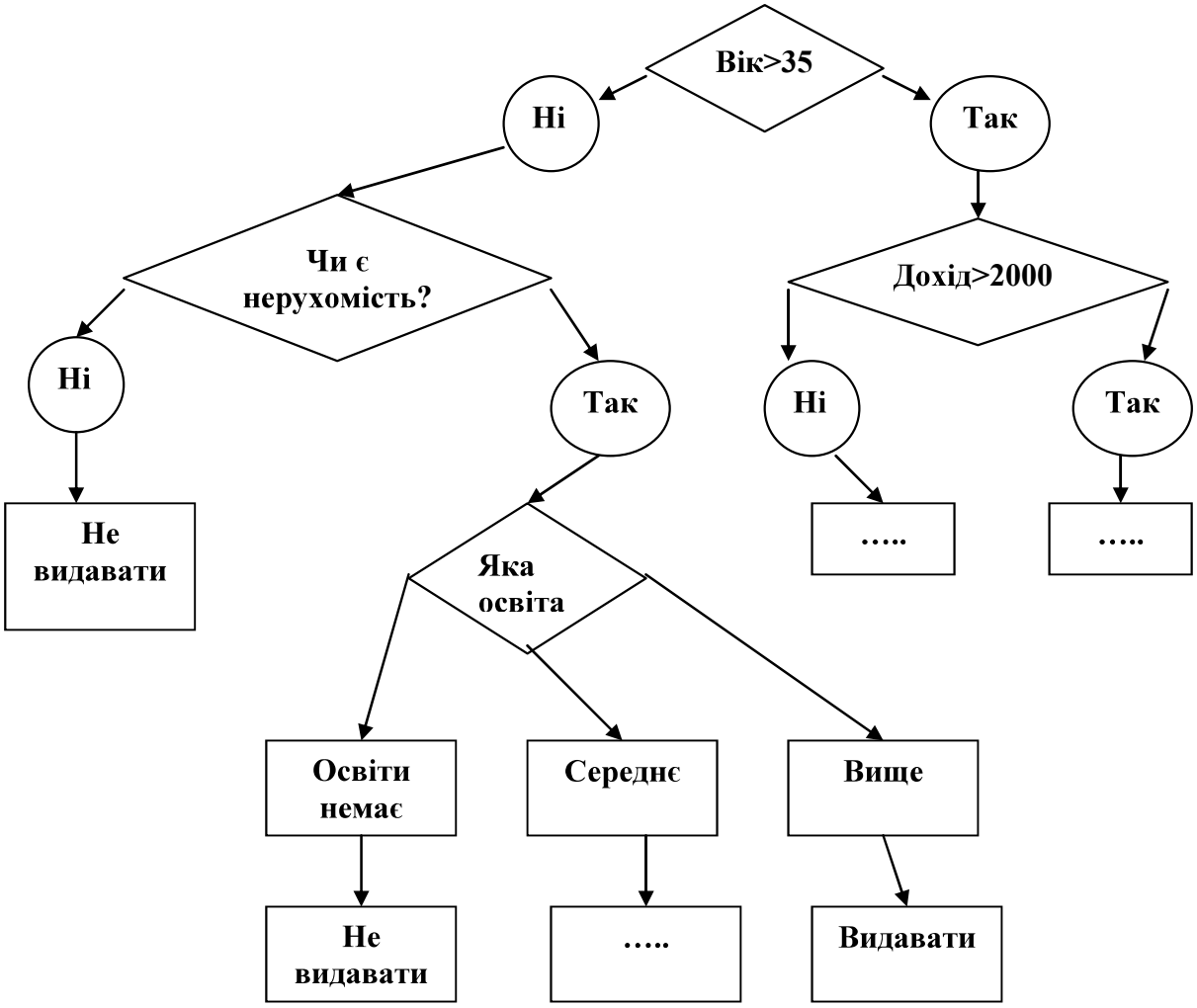


Рис. 4.33. Дерево рішень «Чи видавати кредит?».

Слід зазначити, що на рис. 4.33 змальоване лише одне з можливих дерев рішень для даної бази даних. Наприклад, критерій розщеплювання «Яка освіта?», міг би мати два предикати розщеплювання і виглядати інакше: освіта «вища» і «не вища». Тоді дерево рішень мало б інший вигляд. Таким чином, для даної задачі (як і для будь-якої іншої) може бути побудована множина дерев рішень різної якості, з різною прогнозуючою точністю.

Якість побудованого дерева рішення вельми залежить від правильного вибору критерію розщеплювання. Над розробкою і удосконаленням критеріїв працюють багато дослідників. Метод дерев рішень часто називають «наївним»

підходом. Але завдяки цілому ряду переваг, даний метод є одним з найбільш популярних для вирішення задач класифікації.

Розглянемо процес конструювання дерева рішень. Хай нам задана деяка навчальна множина T, що містить об'єкти (приклади), кожен з яких характеризується m атрибутами, причому один з них вказує на приналежність об'єкту до певного класу. Ідею побудови дерев рішень з множини T, вперше висловлену Хантом, приведемо по Р. Куїнлену (R. Quinlan).

Хай через $\{C_1, C_2, ..., C_k\}$ позначені класи (значення мітки класу), тоді існують 3 ситуації:

1. множина T містить один або більше прикладів, що відносяться до одного класу C_k. Тоді дерево рішень для T – це лист, що визначає клас C_k;

2. множина T не містить жодного прикладу, тобто порожня множина. Тоді це знову лист, і клас, що асоціюється з листом, вибирається з іншої множини відмінної від T, скажімо, з множини, що асоціюється з родителем;

3. множина T містить приклади, що відносяться до різних класів. В цьому випадку слід розбити множину T на деякі підмножини. Для цього вибирається одна з ознак, що має два і більше відмінних один від одного значень $O_1, O_2, ..., O_n$. Множина T розбивається на підмножини $T_1, T_2, ..., T_n$, де кожна підмножина T_i містить всі приклади, що мають значення O_i для вибраної ознаки. Це процедура рекурсивно продовжуватиметься до тих пір, поки кінцева множина не складатиметься з прикладів, що відносяться до одного і тому ж класу.

Вищеописана процедура лежить в основі багатьох сучасних алгоритмів побудови дерев рішень, цей метод відомий ще під назвою *розділення і захвату* (divide and conquer). Вочевидь, що при використанні даної методики, побудова дерева рішень буде відбувається зверху вниз. Оскільки всі об'єкти були заздалегідь віднесені до відомих нам класів, такий процес побудови дерева рішень називається *навчанням з вчителем* (supervised learning). Процес навчання також називають *індуктивним навчанням* або індукцією дерев (tree induction).

Алгоритми конструювання дерев рішень складаються з етапів «побудови» або «створення» дерева (tree building) і «скорочення» дерева (tree pruning). В ході створення дерева вирішуються питання вибору критерію розщеплювання і зупинки навчання (якщо це передбачено алгоритмом). В ході етапу скорочення дерева вирішується питання відсікання деяких його гілок. Розглянемо ці питання детальніше.

Критерій розщеплювання. Процес створення дерева відбувається зверху вниз, тобто є низхідним. В ході процесу алгоритм повинен знайти такий критерій розщеплювання, інколи також званий критерієм розбиття, аби розбити множину на підмножини, які б асоціювалися з даним вузлом перевірки. Кожен вузол перевірки має бути помічений певним атрибутом. Існує правило вибору атрибуту: він повинен розбивати вихідну множину даних так, щоб об'єкти підмножин, що отримуються в результаті цього розбиття, були представниками одного класу або ж були максимально наближені до такого розбиття. Остання фраза означає, що кількість об'єктів з інших класів, так званих «домішок», в кожному класі повинно прагнути до мінімуму.

Існують різні критерії розщеплювання. Найбільш відомі - міра ентропії і індекс Gini.

У деяких методах для вибору атрибуту розщеплювання використовується так звана міра інформативності підпросторів атрибутів, яка ґрунтується на підході ентропії і відома під назвою «міра інформаційного виграшу» (information gain measure) або міра ентропії.

$$Gain(X) = Info(T) - Info_X(T),$$

де $Info(T)$ - ентропія множини T, а $Info_X(T) = \sum_{i=1}^{n} \frac{|T_i|}{|T|} * Info(T_i)$.

Множини $T_1, T_2, ..., T_n$ отримані при розбитті вихідної множини T по перевірці X. Вибирається атрибут, що дає максимальне значення по критерію. Вперше цей захід був запропонований Р. Куїнленом в розробленому їм алгоритмі ID3.

Інший критерій розщеплювання, запропонований Брейманом (Breiman), реалізовано в алгоритмі CART і називається індексом Gini (на честь

італійського економіста Corrado Gini). За допомогою цього індексу атрибут вибирається на підставі відстаней між розподілами класів. Якщо дана множина T, що включає приклади з n класів, індекс Gini, тобто gini(T), визначається по формулі:

$$gini(T) = 1 - \sum_{j=1}^{n} p_j^2,$$

де T - поточний вузол, p_j - імовірність класу j у вузлі T, n - кількість класів.

Велике дерево не означає, що воно «підходяще». Чим більше окремих випадків описано в дереві рішень, тим менша кількість об'єктів потрапляє в кожен окремий випадок. Такі дерева називають «гіллястими» або «кущистими», вони складаються з невиправдано великого числа вузлів і гілок, вихідна множина розбивається на велике число підмножин, що складаються з дуже малого числа об'єктів. В результаті «переповнювання» таких дерев їх здібність до узагальнення зменшується, і побудовані моделі не можуть давати вірні відповіді.

В процесі побудови дерева, аби його розміри не стали надмірно великими, використовують спеціальні процедури, які дозволяють створювати оптимальні дерева, так звані дерева «відповідних розмірів». Який розмір дерева може вважатися оптимальним? Дерево має бути достатнє складним, аби враховувати інформацію з досліджуваного набору даних, але одночасно воно має бути достатнє простим. Іншими словами, дерево повинне використовувати інформацію, поліпшуючу якість моделі, і ігнорувати ту інформацію, яка її не покращує. Тут існує дві можливі стратегії. Перша полягає в нарощуванні дерева до певного розміру відповідно до параметрів, заданих користувачем. Визначення цих параметрів може ґрунтуватися на досвіді і інтуїції аналітика, а також на деяких «діагностичних повідомленнях» системи, що конструює дерево рішень. Друга стратегія полягає у використанні набору процедур, що визначають «відповідний розмір» дерева, вони розроблені Бріманом, Куїлендом і ін. в 1984 році. Проте, як відзначають автори, не можна сказати, що ці процедури доступні початкуючому користувачеві. Процедури, які

використовують для запобігання створення надмірно великих дерев, включають: скорочення дерева шляхом відсікання гілок; використання правил зупинки навчання. Слід зазначити, що не всі алгоритми при конструюванні дерева працюють за однією схемою. Деякі алгоритми включають два окремі послідовні етапи: побудова дерева і його скорочення; інші чергують ці етапи в процесі своєї роботи для запобігання нарощуванню внутрішніх вузлів.

Зупинка побудови дерева. Розглянемо правило зупинки. Воно повинне визначити, чи є даний вузол внутрішнім вузлом, при цьому він розбиватиметься далі, або ж він є кінцевим вузлом, тобто вузлом рішенням.

Означення. Зупинка - такий момент в процесі побудови дерева, коли слід припинити подальші галуження.

Один з варіантів правил зупинки використовує статистичні методи для оцінки доцільності подальшого розбиття – «рання зупинка» (prepruning), вона визначає доцільність розбиття вузла. Перевага використання такого варіанту - зменшення часу на навчання моделі. Проте тут виникає ризик зниження точності класифікації. Тому рекомендується замість зупинки використовувати відсікання.

Другий варіант зупинки навчання - обмеження глибини дерева. В цьому випадку побудова закінчується, якщо досягнута задана глибина. Ще один варіант зупинки - завдання мінімальної кількості прикладів, які міститимуться в кінцевих вузлах дерева. При цьому варіанті галуження тривають до того моменту, поки всі кінцеві вузли дерева не будуть чистими або міститимуть не більше ніж задане число об'єктів. Існує ще ряд правил, але слід зазначити, що жодне з них не має великої практичної цінності, а деякі можуть застосовуватися лише в окремих випадках.

Скорочення дерева або відсікання гілок. Дуже часто алгоритми побудови дерев рішень дають складні дерева, які мають багато вузлів і гілок. Такі дерева дуже важко зрозуміти. До того ж гіллясте дерево, що має багато вузлів, розбиває навчальну множину на досить велику кількість підмножин, що складаються зі все меншої кількості об'єктів. Цінність правила, при цьому,

вкрай низьке, і в цілях аналізу даних таке правило практично непридатне. Багато важливіше мати дерево, що складається з малої кількості вузлів, яким би відповідала велика кількість об'єктів з навчальної вибірки. І тут виникає питання: а чи не побудувати всі можливі варіанти дерев, які відповідають навчальній множині, і з них вибрати дерево з найменшою глибиною? На жаль, ця задача є NP-повною, і, як відомо, цей клас задач не має ефективних методів рішення.

Вирішенням проблеми дуже гіллястого дерева є його скорочення шляхом відсікання (pruning) деяких гілок. Якість класифікаційної моделі, побудованої за допомогою дерева рішень, характеризується двома основними ознаками: точністю розпізнавання і помилкою. Точність розпізнавання розраховується як відношення об'єктів, правильно класифікованих в процесі навчання, до загальної кількості об'єктів набору даних, які брали участь в навчанні. Помилка розраховується як відношення об'єктів, неправильно класифікованих в процесі навчання, до загальної кількості об'єктів набору даних, які брали участь в навчанні.

Відсікання гілок або заміну деяких гілок піддеревом слід проводити там, де ця процедура не приводить до зростання помилки. Процес проходить від низу до верху. Це популярніша процедура, ніж використання правил зупинки. Дерева, що отримуються після відсікання деяких гілок, називають усіченими. Якщо таке усічене дерево все ще не є інтуїтивним і складно для розуміння, використовують витягання правил, які об'єднують в набори для опису класів. Кожна дорога від кореня дерева до його вершини або листа дає одне правило. Умовами правила є перевірки на внутрішніх вузлах дерева. Хоча відсікання не є панацеєю, але в більшості практичних задач дає добрі результати, що дозволяє говорити про правомірність використання подібної методики (рис. 4.34).

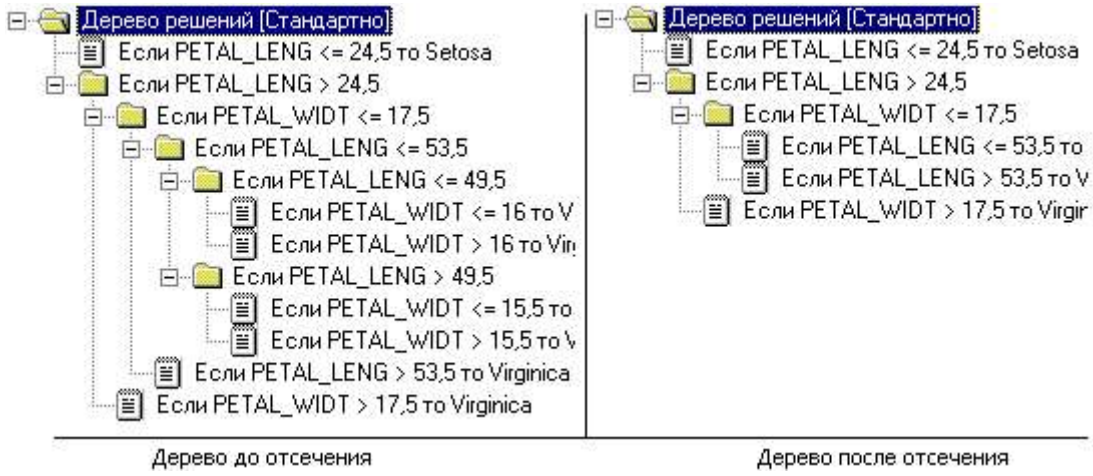


Рис. 4.34. Процедура відсікання деяких гілок.

У користувачів систем інтелектуального аналізу даних досить часто виникає питання: чи є якась суттєва відмінність алгоритму дерева рішень від асоціативних правил в задачах класифікації. Нагадаємо, що алгоритм асоціативних правил призначений для виявлення в даних залежностей типа $A \geq B$, де A і B є наборами пар атрибут = значення. Алгоритм дерева рішень призначений для вирішення задач регресії і класифікації, тобто для виявлення залежності цільового параметра від значення інших параметрів. Так само, як і у випадку з алгоритмом асоціативних правил, алгоритм дерева рішень дозволяє виявляти правила в даних типа $A \geq B$. Для цього в якості лівої частини правила (умови А) треба об'єднати умови по всіх вузлах у вітці дерева від кореня до листа, що містить опис умов на цільову змінну В.

Не дивлячись на схожість вирішуваної задачі і форму представлення результату, правила, отримувані обома алгоритмами, можуть істотно розрізнятися. Вся річ у тому, що процес побудови дерева рішень заснований на максимізації приросту інформації, тоді як алгоритм асоціативних правил заснований на виділенні частих наборів, в яких частота появи однієї частини (права частина правила) відносно наборів, в яких є інша частина (ліва частина правила), висока. Виходячи з приведеного аналізу відмінності між алгоритмами, ми можемо побудувати приклад, що наводить до принципово різних результатів.

Передбачимо, що магазин містить в асортименті три товари: товар А, товар Б і товар В Допустимо, що ми хочемо виділити правила, які передбачають факт покупки товару А. Побудуємо дерево рішень. Зі всіх транзакцій (10000) одна частина (6000) містить товар А, а інша (4000) не містить. Далі алгоритм дерева рішень повинен перевірити розбиття транзакцій за фактом наявності товару Б і наявність товару В. Кінцеве розбиття буде здійснено по атрибуту, що найбільшою мірою збільшує приріст інформації за фактом покупки товару А.

Розглянемо спочатку можливість розбиття за фактом покупки товару Б. Передбачимо, що товар Б міститься в 8000 транзакцій. У одній частині цих транзакцій (4800) міститься товар А, а в іншій (3200) - ні. Далі, в 2000 транзакціях з тих, що не містять товар Б, частина (1200) містить товар А, а частина (800) - ні. Варіант дерева рішень має слідуючий вигляд (рис. 4.35):

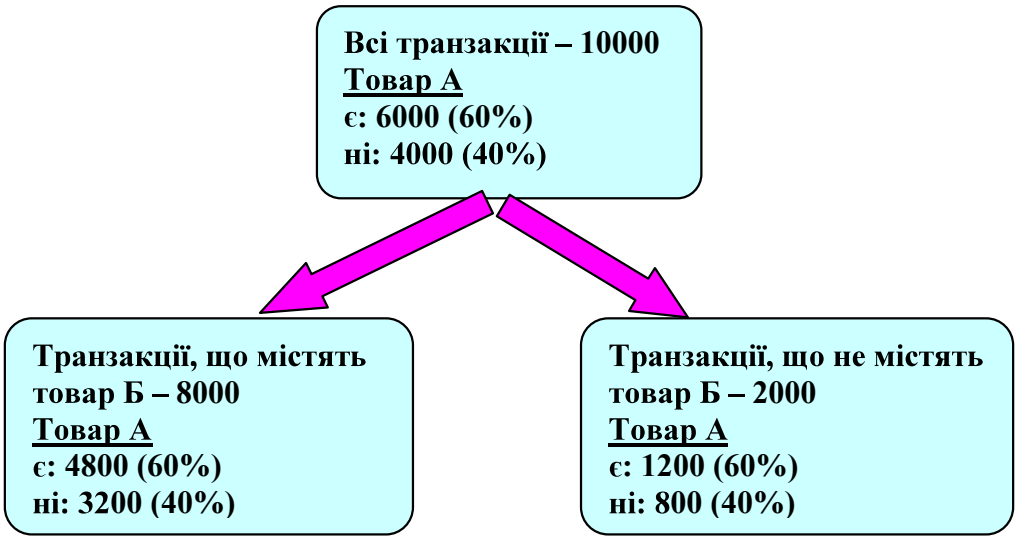


Рис. 4.35. Дерево рішень факту покупки товару А (варіант 1).

Таким чином, розбиття по наявності товару Б в транзакції не збільшує приріст інформації (розподіл транзакцій, що містять товар А зберігається після розбиття за фактом покупки товару Б). Тобто алгоритм дерева рішень не здійснюватиме розбиття за ознакою наявності товару Б в транзакції. При цьому,

алгоритм асоціативних правил швидше за все сформулює правило вигляду товар Б ≥ товар А, оскільки транзакції, що містять одночасно товар А і товар Б є дуже частими (4800 або 48% транзакцій), а точність дотримання правила складає 60%, що часто входить в порогове значення аналізованих правил в алгоритмі асоціативних правил.

Тепер розглянемо варіант побудови дерева рішень з розбиттям за фактом покупки товару В. Передбачимо, що 1000 транзакцій містить товар В, а ті 9000, що залишилися - ні. З 1000 транзакцій, що містять товар В, 900 транзакцій містить товар А, а 100 - ні. З 9000 транзакцій, що не містять товар В, 5100 транзакцій містять товар А, а 3900 - ні. Нижче побудоване відповідне дерево рішень (рис. 4.36).

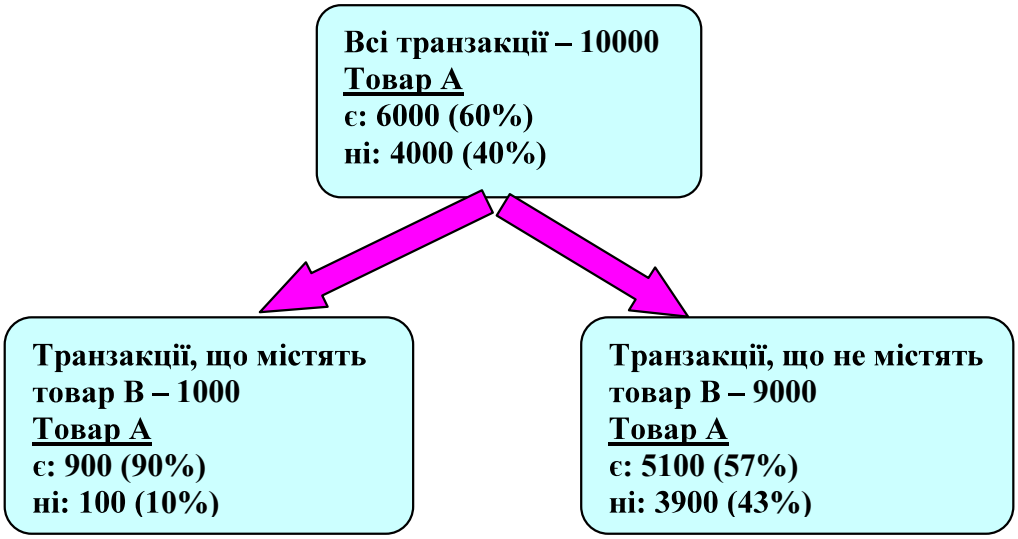


Рис. 4.36. Дерево рішень факту покупки товару А (варіант 2).

Розбиття за ознакою наявності товару В дає в разі позитивної відповіді значний приріст інформації: 90% транзакцій, що містять товар В, містять товар А, тобто дерево рішень сформулює правило товар В ≥ товар А. Алгоритм асоціативних правил може не дати аналогічного правила, оскільки транзакції, що одночасно містять товари А і В, складають лише 9% від загального числа транзакцій, що може не пройти поріг підтримки правила в алгоритмі.

Таким чином, на даних сконструйованого прикладу, алгоритм дерева рішень дасть правило товар В ≥ товар А, а алгоритм асоціативних правил - товар Б ≥ товар А. Вибір алгоритму виявлення правил в даних може істотно вплинути на результат аналізу. Можлива відмінність в результатах пов'язана з особливістю реалізації кожного з описаних алгоритмів і не є помилкою одного з них. В зв'язку з цим має сенс рекомендувати їх спільне вживання і використання експертного знання для вибору найбільш адекватного для кожної конкретної моделі.

Розглянувши основні проблеми, що виникають при побудові дерев, було б несправедливо не згадати про їх достоїнства.

1. *Швидкий процес навчання*. На побудову класифікаційних моделей за допомогою алгоритмів конструювання дерев рішень потрібно значно менше часу, чим, наприклад, на навчання нейронних мереж. Більшість алгоритмів конструювання дерев рішень мають можливість спеціальної обробки пропущених значень. Багато класичних статистичних методів, за допомогою яких вирішуються задачі класифікації, можуть працювати лише з числовими даними, тоді як дерева рішень працюють і з числовими, і з категоріальними типами даних.

2. *Генерація правил в областях, де експертові важко формалізувати свої знання*. Дерева рішень дозволяють створювати класифікаційні моделі в тих областях, де аналітику досить складно формалізувати знання. Алгоритм конструювання дерева рішень не вимагає від користувача вибору вхідних атрибутів (незалежних змінних). На вхід алгоритму можна подавати всі існуючі атрибути, алгоритм сам вибере найбільш значимі серед них, і лише вони будуть використані для побудови дерева. У порівнянні, наприклад, з нейронними мережами, це значно полегшує користувачеві роботу, оскільки в нейронних мережах вибір кількості вхідних атрибутів істотно впливає на час навчання.

3. *Витягання правил на природній мові*. Дерева рішень дають можливість витягувати правила з бази даних на природній мові. Приклад правила: «Якщо Вік > 35 і Дохід > 200, то видати кредит».

4. *Інтуїтивно зрозуміла класифікаційна модель*. Класифікаційна модель, представлена у вигляді дерева рішень, є інтуїтивною і спрощує розуміння вирішуваного завдання. Результат роботи алгоритмів конструювання дерев рішень, на відмінність, наприклад, від нейронних мереж, що є «чорними ящиками», легко інтерпретується користувачем. Ця властивість дерев рішень не лише важлива при віднесенні до певного класу нового об'єкту, але і корисна при інтерпретації моделі класифікації в цілому. Дерево рішень дозволяє зрозуміти і пояснити, чому конкретний об'єкт відноситься до того або іншого класу.

5. *Висока точність прогнозу, порівняно з іншими методами.* Точність моделей, створених за допомогою дерев рішень значно вища, порівняно з іншими методами побудови класифікаційних моделей (статистичні методи, нейроні мережі та інше). Розроблений ряд масштабованих алгоритмів, які можуть бути використані для побудови дерев рішень на надвеликих базах даних; масштабність тут означає, що із зростанням числа прикладів або записів бази даних час, що витрачається на навчання, тобто побудова дерев рішень, зростає лінійно. Приклади таких алгоритмів: SLIQ, SPRINT.

6. *Побудова непараметричних моделей*. Багато статистичних методів є параметричними, і користувач повинен заздалегідь володіти певною інформацією, наприклад, знати вигляд моделі, мати гіпотезу про вигляд залежності між змінними, передбачати, який вигляд розподілу мають дані. Дерева рішень, на відміну від таких методів, будують непараметричні моделі. Таким чином, дерева рішень здатні вирішувати такі задачі інтелектуального аналізу даних, в яких відсутня апріорна інформація про вигляд залежності між досліджуваними даними.

Через ці і багато інших причин, методологія дерев рішень є важливим інструментом в роботі кожного фахівця, що займається аналізом даних, незалежно від того практик він або теоретик.

На сьогоднішній день існує значне число алгоритмів, що реалізовують дерева рішень CART, C4.5, NewId, ITrule, CHAID, CN2 і так далі Але найбільше поширення і популярність отримали наступні два:

- CART (Classification and Regression Tree) – це алгоритм побудови бінарного дерева рішень – дихотомічної класифікаційної моделі. Кожен вузол дерева при розбитті має лише двох нащадків. Як видно з назви алгоритму, він вирішує задачі класифікації і регресії.
- C4.5 – алгоритм побудови дерева рішень, у якого кількість нащадків у вузла не обмежена. Алгоритм вміє працювати з безперервним цільовим полем, тому вирішує лише задачі класифікації.

Алгоритм CART. CART, скорочення від Classification And Regression Tree, перекладається як «Дерево класифікації і регресії» – алгоритм бінарного дерева рішень. Він був розроблений в 1974-1984 роках чотирма професорами статистики - Leo Breiman (Berkeley), Jerry Friedman (Stanford), Charles Stone (Berkeley) і Richard Olshen (Stanford). Алгоритм призначений для вирішення задач класифікації і регресії. Існує також декілька модифікованих версій – алгоритми IndCART і DB-CART. Алгоритм IndCART, є частиною пакету Ind і відрізняється від CART використанням іншого способу обробки пропущених значень, не здійснює регресійну частину алгоритму CART і має інші параметри відсікання. Алгоритм DB-CART базується на наступній ідеї: замість того аби використовувати навчальний набір даних для визначення розбиття, використовуємо його для оцінки розподілу вхідних і вихідних значень і потім використовуємо цю оцінку, аби визначити розбиття. DB, відповідно означає – «Distribution based». Стверджується, що ця ідея дає значне зменшення помилки класифікації, в порівнянні із стандартними методами побудови дерева. Основними відмінностями алгоритму CART від алгоритмів сімейства ID3 є:

- бінарне представлення дерева рішень;
- функція оцінки якості розбиття;
- механізм відсікання дерева;
- алгоритм обробки пропущених значень;

- побудова дерев регресії.

Бінарне представлення дерева рішень. У алгоритмі CART кожен вузол дерева рішень має двох нащадків. На кожному кроці побудови дерева правило, яке формується у вузлі, ділить задану множину прикладів (навчальну вибірку) на дві частини – частину, в якій виконується правило (нащадок – right) і частину, в якій правило не виконується (нащадок – left). Для вибору оптимального правила використовується функція оцінки якості розбиття.

Функція оцінки якості розбиття. Навчання дерева рішень відноситься до класу навчання з вчителем, тобто навчальна і тестова вибірки містять класифікований набір прикладів. Оціночна функція, використовувана алгоритмом CART, базується на інтуїтивній ідеї зменшення нечистоти (невизначеності) у вузлі. Розглянемо задачу з двома класами і вузлом, що має по 50 прикладів одного класу. Вузол має максимальну «нечистоту». Якщо буде знайдено розбиття, яке розбиває дані на дві підгрупи 40:5 прикладів в одній і 10:45 в іншій, то інтуїтивно «нечистота» зменшиться. Вона повністю зникне, коли буде знайдено розбиття, яке створить підгрупи 50:0 і 0:50. У алгоритмі CART ідея «нечистоти» формалізована в індексі Gini. Якщо набір T розбивається на дві частини T_1 і T_2 з числом прикладів в кожному N_1 і N_2 відповідно, тоді показник якості розбиття буде рівний

$$Gini_{split}(T) = \frac{N_1}{N} Gini(T_1) + \frac{N_2}{N} Gini(T_2).$$

Найкращим вважається те розбиття, для якого $Gini_{split}(T)$ мінімально.

Позначимо N – число прикладів у вузлі – предку, L, R – число прикладів відповідно в лівому і правому нащадку, l_i і r_i – число екземплярів i класу в лівому/правому нащадку. Тоді якість розбиття оцінюється по наступній формулі

$$Gini_{split} = \frac{L}{N}\left(1 - \sum_{i=1}^{n}\left(\frac{l_i}{L}\right)^2\right) + \frac{R}{N}\left(1 - \sum_{i=1}^{n}\left(\frac{r_i}{R}\right)^2\right) \to \min.$$

Аби зменшити об'єм обчислень формулу можна перетворити до вигляду

$$Gini_{split}(T) = \frac{1}{L}\sum_{i=1}^{n} l_i^2 + \frac{1}{R}\sum_{i=1}^{n} r_i^2 \to \max .$$

В результаті, кращим буде те розбиття, для якого величина максимальна. Рідше в алгоритмі CART використовуються інші критерії розбиття Twoing, Symmetric Gini і ін.

Правила розбиття. Вектор предикторних змінних, що подається на вхід дерева може містити як числові (порядкові) так і категоріальні змінні. В будь-якому разі в кожному вузлі розбиття йде лише по одній змінній. Якщо змінна числового типа, то у вузлі формується правило вигляду $x_i \le c$. Де c – деякий поріг, який найчастіше вибирається як середнє арифметичне двох сусідніх впорядкованих значень змінної x_i навчальної вибірки. Якщо змінна категоріального типа, то у вузлі формується правило $x_i * V(x_i)$, де $V(x_i)$ – деяка непорожня підмножина множини значень змінної x_i в навчальній вибірці. Отже, для n значень числового атрибуту алгоритм порівнює $n-1$ розбиття, а для категоріального $(2^{n-1}-1)$. На кожному кроці побудови дерева алгоритм послідовно порівнює все можливе розбиття для всіх атрибутів і вибирає найкращий атрибут і найкраще розбиття для нього.

Механізм відсікання дерева. Механізм відсікання дерева, оригінальна назва minimal cost-complexity tree pruning, – найбільш серйозна відмінність алгоритму CART від інших алгоритмів побудови дерева. CART розглядає відсікання як здобуття компромісу між двома проблемами: здобуття дерева оптимального розміру і здобуття точної оцінки вірогідності помилкової класифікації.

Основна проблема відсікання – велика кількість всіх можливих відсічених піддерев для одного дерева. Точніше, якщо бінарне дерево має $|T|$ – листів, тоді існує ~$[1.5028369^{|T|}]$ відсічених піддерев. І якщо дерево має хоч би 1000 листів, тоді число відсічених піддерев стає просто величезним. Базова ідея методу – не розглядати всі можливі піддерева, обмежившись лише «кращими представниками» згідно приведеній нижче оцінці.

Позначимо $|T|$ – число листів дерева, $R(T)$ – помилка класифікації дерева, рівна відношенню числа неправильно класифікованих прикладів до прикладів в навчальній вибірці. Визначимо $C_\alpha(T)$ – повну вартість (оцінку/показник витрати-складність) дерева T як

$$C_\alpha(T) = R(T) + \alpha|T|,$$

де $|T|$ – число листів (термінальних вузлів) дерева, α – деякий параметр, що змінюється від 0 до $+\infty$. Повна вартість дерева складається з двох компонент – помилки класифікації дерева і штрафу за його складність. Якщо помилка класифікації дерева незмінна, тоді із збільшенням повна вартість дерева збільшуватиметься. Тоді залежно від неї гіллясте дерево, що дає більшу помилку класифікації може коштувати менше, ніж те, що дає меншу помилку, але більш гіллясте.

Визначимо $T_{\max}$ – максимальне за розміром дерево, яке належить обрізувати. Якщо ми зафіксуємо значення α, тоді існує найменше мінімізуєме піддерево α, яке виконує наступні умови

$$C_\alpha(T(\alpha)) = \min_{T \le T_{\max}} C\alpha(T),$$

$$if C_\alpha(T) = C_\alpha(T(\alpha)) then T(\alpha) \le T.$$

Перша умова говорить, що не існує такого піддерева дерева $T_{\max}$, яке мало б меншу вартість, чим $T(\alpha)$ при цьому значенні α. Друга умова говорить, що якщо існує більш за одне піддерево, яке має таку ж повну вартість, тоді ми вибираємо найменше дерево.

Вибір фінального дерева. Отже, якщо ми маємо послідовність дерев, то нам необхідно вибрати краще дерево з неї. Найбільш очевидним є вибір фінального дерева через тестування на тестовій вибірці. Дерево, що дало мінімальну помилку класифікації, і буде кращим. Проте, це не єдино можлива дорога.

Перехресна перевірка. Перехресна перевірка (V-fold cross-validation) – найоригінальніша і складніша частина алгоритму CART. Цей шлях вибору фінального дерева використовується, коли набір даних для навчання малий або

кожен запис в ньому по своєму унікальний так, що ми не можемо виділити вибірку для навчання і вибірку для тестування. В такому разі будуємо дерево на всіх даних, обчислюваний $\alpha_1,\alpha_2,...,\alpha_k$ і $T_1 > T_2 > ... > T_n$. Позначимо T_k – найменше піддерево, що мінімізується, для (α_k,α_{k+1}). Тепер ми хочемо вибрати дерево з послідовності, але вже використали все наявні дані. Хитрість в тому, що ми збираємося обчислити помилку дерева T_k з послідовності непрямим шляхом.

Крок 1. Встановимо $\beta_1 = 0, \beta_2 = \sqrt{\alpha_2\alpha_3}, \beta_3 = \sqrt{\alpha_3\alpha_4},..., \beta_{N-1} = \sqrt{\alpha_{N-1}\alpha_N}, \beta_N = \infty$. Вважається, що β_k буде типовим значенням для (α_k,α_{k+1}) і, отже, як значення відповідає T_k.

Крок 2. Розділимо весь набір даних на V груп однакового розміру $G_1,G_2,...,G_V$. Рекомендується брати $V = 10$. Потім для кожної групи G_i:

1. Обчислити послідовність дерев за допомогою описаного вище механізму відсікання на всіх даних, виключаючи G_i. І визначити $T^{(i)}(\beta_1),...,T^{(i)}(\beta_N)$ для цієї послідовності.

2. Обчислити помилку дерева $T^{(i0}(\beta_k)$ на G_i. Тут $T^{(i0}(\beta_k)$ означає найменше мінімізоване піддерево з послідовності, побудоване на всіх даних, виключаючи G_i.

Крок 3. Для кожного β_k підсумовувати помилку $T^{(i0}(\beta_k)$ по всіх G_i. Хай β_h буде з найменшою загальною помилкою. Оскільки β_h відповідає дереву T_h, ми вибираємо T_h з послідовності побудованої на всіх даних як фінальне дерево. Показник помилки, обчислений за допомогою перехресної перевірки можна використовувати як оцінку помилки дерева.

Алгоритм обробки пропущених значень. Більшість алгоритмів інтелектуального аналізу даних передбачають відсутність пропущених значень. У практичному аналізі це припущення часто є невірним. Найбільш загальне рішення – відкинути дані, які містять один або декілька порожніх атрибутів. Проте це рішення має свої недоліки:

- Зсув даних. Якщо викинуті дані лежать декілька в стороні від залишених, тоді аналіз може дати упереджені результати.
- Зменшення потужності. Може виникнути ситуація, коли доведеться викинути багато даних. В такому разі точність прогнозу сильно зменшується.

Якщо ми хочемо будувати і використовувати дерево на неповних даних нам необхідно вирішити наступні питання - як визначити якість розбиття та у яку гілку необхідно послати спостереження, якщо пропущена змінна є такою, на яку доводиться найкраще розбиття.

Аби визначити якість розбиття CART просто ігнорує пропущені значення. Це вирішує першу проблему, але ми ще повинні вирішити, по якій дорозі посилати спостереження з пропущеною змінною що містить найкраще розбиття. З цією метою CART обчислює так зване «сурогатне» розбиття. Воно створює найбільш близькі до кращого підмножини прикладів в поточному вузлі.

Регресія. Побудова дерева регресії багато в чому схожа з деревом класифікації. Спочатку будується дерево максимального розміру, потім обрізується дерево до оптимального розміру.

Основне достоїнство дерев в порівнянні з іншими методами регресії – можливість працювати з багатовимірними задачами і задачами, в яких присутня залежність вихідний змінної від змінної або змінних категоріального типа. Основна ідея – розбиття всього простору на прямокутники, необов'язкового однакового розміру, в яких вихідна змінна вважається постійною (рис. 4.37).

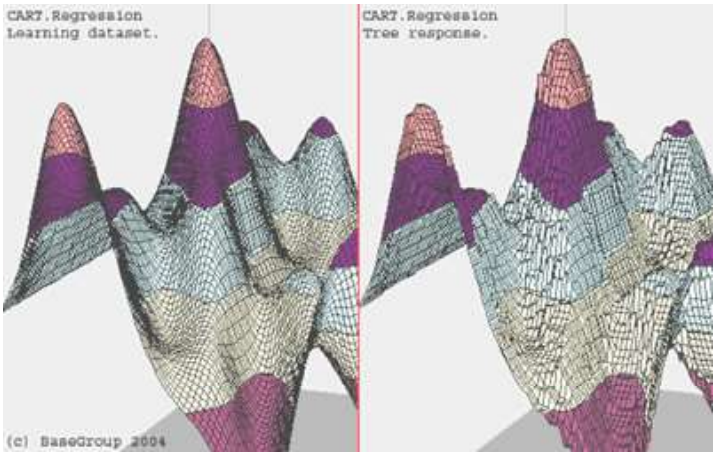


Рис. 4.37. Приклад двовимірної задачі.

Процес побудови дерева відбувається послідовно. На першому кроці отримуємо регресійну оцінку просто як константу по всьому простору прикладів. На другому кроці ділимо простір на дві частини. Якщо ми позначимо всі значення вихідний змінної як $Y_1, Y_2, ..., Y_n$ тоді регресійна оцінка має вигляд:

$$\bar{f}(x) = \left(\frac{\mathbb{1}}{n} \sum_{i=1}^{n} Y_i \right) I_R(x),$$

де R – простір навчальних прикладів, n – число прикладів, $I_R(x)$ – індикаторна функція простору – фактично, набір правил, що описують попадання змінної x в простір.

Алгоритм C4.5. Алгоритм C4.5 будує дерево рішень з необмеженою кількістю гілок у вузла. Даний алгоритм може працювати лише з дискретним залежним атрибутом і тому може вирішувати лише задачі класифікації. C4.5 вважається одним з найвідоміших і широко використовуваних алгоритмів побудови дерев класифікації, який вперше був запропонований Р. Куїнленом (R. Quinlan). Перш ніж приступити до опису алгоритму побудови дерева рішень, визначимо обов'язкові вимоги до структури даних і безпосередньо до самих даних, при виконанні яких алгоритм C4.5 буде працездатний:

- Опис атрибутів. Дані, необхідні для роботи алгоритму, мають бути представлені у вигляді плоскої таблиці. Вся інформація про об'єкти з предметної області повинна описуватися у вигляді кінцевого набору ознак (атрибутів). Кожен атрибут повинен мати дискретне або числове значення. Самі атрибути не повинні мінятися від прикладу до прикладу, і кількість атрибутів має бути фіксованою для всіх прикладів.
- Певні класи. Кожен приклад має бути асоційований з конкретним класом, тобто один з атрибутів має бути вибраний як мітка класу.
- Дискретні класи. Класи мають бути дискретними, тобто мати кінцеве число значень. Кожен приклад повинен однозначно відноситися до конкретного класу. Випадки, коли приклади належать до класу з імовірнісними оцінками, виключаються. Кількість класів має бути значно менше кількості прикладів.

Розглянемо алгоритм побудови дерева. Хай нам задана множина прикладів T, де кожен елемент цієї множини описується m атрибутами. Кількість прикладів в множині T називатимемо потужністю цієї множини і позначатимемо $|T|$. Хай мітка класу набуває наступних значень $C_1, C_2, ..., C_k$. Задача полягатиме в побудові ієрархічної класифікаційної моделі у вигляді дерева з множиною прикладів T. Процес побудови дерева відбуватиметься зверху вниз. Спочатку створюється корінь дерева, потім нащадки кореня і так далі

На першому кроці ми маємо порожнє дерево (є лише корінь) і вихідну множину T (асоційовану з коренем). Потрібно розбити вихідну множину на підмножини. Це можна зробити, вибравши один з атрибутів як перевірку. Тоді в результаті розбиття виходять n (по числу значень атрибуту) підмножин і, відповідно, створюються n нащадків кореня, кожному з яких поставлена у відповідність своя підмножина, отримана при розбитті множини T. Потім ця процедура рекурсивно застосовується до всіх підмножин (нащадкам кореня) і так далі.

Отже, ми маємо дерево рішень і хочемо використовувати його для розпізнавання нового об'єкту. Обхід дерева рішень починається з кореня дерева. На кожному внутрішньому вузлі перевіряється значення об'єкту Y по атрибуту, який відповідає перевірці в даному вузлі, і, залежно від отриманої відповіді, знаходиться відповідне галуження, і по цій дузі рухаємося до вузла, що знаходить на рівень нижче і так далі Обхід дерева закінчується як тільки зустрінеться вузол рішення, який і дає назву класу об'єкту Y.

Остання версія алгоритму - алгоритм C4.8 - реалізована в інструменті Weka як J4.8 (Java), комерційна реалізація методу - C5.0, розробник RuleQuest, Австралія. Слід зазначити, що алгоритм C4.5 повільно працює на надвеликих і зашумлених наборах даних.

Найбільш серйозна вимога, яка зараз пред'являється до алгоритмів конструювання дерев рішень, - це масштабність, тобто алгоритм повинен володіти масштабованим методом доступу до даних. Розроблений ряд нових

масштабованих алгоритмів, серед них - алгоритм Sprint, запропонований Джоном Шафером і його колегами. Sprint, що є масштабованим варіантом розглянутого алгоритму CART, пред'являє мінімальні вимоги до об'єму оперативної пам'яті.

4.5. Комп’ютерні системи та напрямки застосування дерев рішень.

Дерева рішень корисні для бізнес-користувачів, оскільки надають логічний результат, який можна обговорювати в бізнес-термінах. Вони просто створюються із застосуванням різних програмних рішень, і дуже ефективні і точні при забезпеченні хорошим набором даних. Практично всі відомі виробники комп'ютерних систем включають до складу своїх програмних продуктів алгоритми побудови і аналізу дерев рішень. Розглянемо деякі з них.

Скорингові системи. Підвищення прибутковості кредитного портфеля банку безпосередньо залежить від грамотного управління кредитними ризиками. І саме скорингові системи дозволяють понизити ризики без втрати прибутковості, запропонувавши відповідь на ключові питання: наскільки проблематичною буде робота банку з конкретним позичальником, яке значення кредитного ліміту встановити, і поверне клієнт кредит чи ні.

У 1941 р. Девід Дюран вперше застосував методику класифікації кредитів на «хороших» і «поганих». Він визначив не лише групи чинників, що дозволяють максимально визначити міру кредитного ризику, але і коефіцієнти, що характеризують кредитоспроможність приватного клієнта. Таким чином, позичальник, який здолав порогове значення, набравши достатню кількість балів, потенційно міг отримати запрошувану суму. Ідея Дюрана отримала продовження - незабаром в Сан-Франциско утворилася перша консалтингова фірма в області скоринга Fair Issac, а декілька пізніше, з появою нових масових кредитних продуктів (кредитних карт), до ідеї скоринга звернулися всі фінансові установи США. По суть, скоринг є методом класифікації сукупності

позичальників на різні групи, коли необхідна характеристика не відома, проте, відомі інші характеристики, які яким-небудь чином корелюють з тією, що цікавить. На практиці, залежно від задач аналізу позичальника, кредитний скоринг включає application-скоринг - оцінку кредитоспроможності претендентів на здобуття кредиту (скоринг за анкетними даними використовується в першу чергу), behavioral-скоринг — оцінка вірогідності повернення виданих кредитів (поведінковий аналіз), а також collection-скоринг - оцінка можливості повного або часткового повернення кредиту при порушенні термінів погашення заборгованості (розрахунок ризиків по портфелю). На сьогодні відомі розробки SAS, KXEN, Experian, SPSS, EGAR, які є не спеціалізованими програмними засобами для скоринга, а універсальними аналітичними інструментами інтелектуального аналізу даних, які можна використовувати для побудови власних скорингових моделей. Тому, в повнішому розумінні, скорингова система зсередини є складною системою автоматизації видачі споживчих кредитів в банківських відділеннях, торгівельних точках, через інтернет, яка як аналітичне ядро використовує рішення однієї з відомих компаній-розробників. Сам по собі скоринг — це не лише робота з певними скоринговими моделями, але і побудова скорингової інфраструктури. Так, в багатьох програмних продуктах результат аналізу статистичних даних (мат. модель) можна зберегти у вигляді програмного коду, а його вставити в банківське програмне забезпечення. Тобто, під скоринговою системою мають на увазі спеціальне програмне забезпечення, за допомогою якого можна розрахувати необхідний показник на основі вихідних даних. Скорингова карта - це набір затверджених банком певних характеристик і відповідних вагових коефіцієнтів (у балах). Скорингових карт в банках зазвичай декілька, оскільки вони сильно залежать від кредитних продуктів. Наприклад, під нерухомість необхідна одна карта, а на покупку автомобіля вже абсолютно інша. На думку експертів, можна використовувати і одну загальну карту, проте це незручно для користувачів. Моделей також майже завжди декілька. Зазвичай заявка на кредит проходить через велику кількість моделей,

причому для різних категорій осіб можуть застосовуватися різні моделі навіть на одній скоринговій карті.

Однією з основних труднощів відомих скорингових систем, як і всіх технологічних рішень у сфері core banking, є їх погана адаптуємість. Річ у тому, що з часом можуть мінятися умови, в яких функціонує позичальник. А значить, скорингові моделі необхідно актуалізувати на найбільш «свіжих» клієнтах, періодично перевіряючи і, при необхідності, розробляючи нову модель, як для різних періодів часу, так і для різних регіонів. Подолання таких обмежень скоринга вирішується за допомогою інструментів інтелектуального аналізу даних. Найбільш поширеним методом автоматичного аналізу даних є побудова дерева рішень. Постачальники рішень упевнені: для того, щоб отримати обґрунтовані висновки, не обов'язково бути статистиком. Наприклад, AnswerTree (продукт SPSS) автоматично будує дерево, дозволяючи на базі діалогових вікон навіть непідготовленому користувачеві почати роботу з програмою. Сама AnswerTree автоматично просіює дані і знаходить статистично значимі групи. За допомогою інтуїтивно зрозумілих деревовидних діаграм, графіків і таблиць програма самостійно сегментує дані, при цьому аналітикові необхідно лише вказати цільову змінну, змінні-предикати і вибрати алгоритм побудови дерева рішень. Зручно, що деревовидна діаграма, яка схожа на блок-схему, дозволяє візуалізувати виділені сегменти і закономірності в даних. Для здобуття максимально достовірних результатів зазвичай рекомендується навчити модель на підвибірці, а потім протестувати надійність на даних, що залишилися. Наскільки добре модель описує дані, можна побачити, перемикаючись з навчальної моделі на контрольну. Представити результати аналізу можна в будь-якому форматі, наприклад, вивести інформацію по кожному вузлу у вигляді таблиці або графіка. Правильно побудоване на даних минулих періодів дерево рішення володіє ще однією дуже важливою особливістю, а саме, «здібністю до узагальнення». Це означає, що якщо виникає нова ситуація, можна з досить великою часткою упевненості

сказати, що позичальник, який знов звернувся, поведеться так само, як і ті позичальники, характеристики яких схожі з характеристиками нових клієнтів.

Скорингова система традиційно складається з модуля підготовки вихідних даних, аналітичного модуля і модуля звітності. Дані системи скоринга, можуть бути трьох типів. Перший тип - знання персоналу кредитних відділів банків про конкретні типи кредитних продуктів і своїх клієнтів. Другий тип даних - статистика по вже виданих кредитах, що враховує «добрих» і «поганих» позичальників. І, якщо банк не володіє жодним з типів вказаних даних - ні експертними знаннями, ні статистикою виданих кредитів, модель, що лежить в основі системи скоринга, переважно будується на основі регіональних і галузевих даних.

Всі фронт-офісні рішення для автоматизації процесу споживчого кредитування в більшості випадків є Web-додатками, що забезпечують хорошу масштабність системи і простоту підключення до процесу видачі кредитів нових відділень банку і представництв в торгівельних точках. Найбільш відомими західними скоринговими системами сьогодні є SAS Credit Scoring, EGAR Scoring, Transact SM (Experian-Scorex), K4Loans (KXEN), Clementine (SPSS). Серед розробників з СНД — BNS, Basegroup Labs. Найбільш серйозними і дорогими є рішення SAS (близько 200 тис. дол.), гідними також вважаються розробки KXEN (близько 30 тис. дол.). Практично посередині цінового діапазону знаходиться пропозиція по скорингу компанії EGAR Technology, яка, з одного боку, є західним вендором, що пропонує скорингову систему, яка використовує класичні західні моделі, з іншого боку, - це рішення (EGAR Scoring) максимально адаптоване до українських умов і доповнене спеціальними підходами - наприклад, макроекономічним підходом до оцінки кредитоспроможності позичальника, обліком особливостей самих кредитних продуктів і іншими можливостями. Порівнюючи західні і вітчизняні системи, необхідно відмітити наступне:

1. Західні системи з'явилися набагато раніше, у них великий термін експлуатації, відповідно великий об'єм кредитних історій, але ці історії не підходять для українського ринку.

2. У західних системах немає інструментів (можливостей) для роботи з малими об'ємами кредитних історій (що необхідне для українського ринку). Розробки в області класичного скоринга дозволяють працювати з обмеженими об'ємами кредитних історій.

3. Ще одна особливість - велика різниця між скоринговими картами залежно від локальних ринків і для різних банківських продуктів. Відповідно, західні системи недостатньо гнучкі для вітчизняного ринку.

Застосування технології дерев рішень для оцінки кредитоспроможності фізичних осіб на основі пакету Deductor. При кредитуванні фізичних осіб характерні невеликі розміри позик, що породжує великий об'єм роботи по їх оформленню і досить дорогу процедуру оцінки кредитоспроможності відносно отримуваного в результаті прибутку. Для оцінки кредитоспроможності фізичних осіб банку необхідно оцінити як фінансове положення позичальника, так і його особисті якості. При цьому кредитний ризик складається з ризику неповернення основної суми боргу і відсотків по цій сумі. Зараз для оцінки ризику кредитування позичальника використовується скоринг кредитування. Сутність цієї методики полягає в тому, що кожен чинник, що характеризує позичальника, має свою кількісну оцінку. Підсумовуючи отримані бали, можна отримати оцінку кредитоспроможності фізичної особи. Кожен параметр має максимально можливий поріг, який вище для важливих питань і нижче для другорядних. На сьогоднішній день відомо досить багато методик кредитного скоринга. Однією з найвідоміших є модель Дюрана. Основним недоліком скорингової системи оцінки кредитоспроможності фізичних осіб є те, що вона дуже погано адаптуєма, а також має високу вартість адаптації використовуваної моделі під поточне положення справ і велику ймовірність помилки моделі при визначенні

кредитоспроможності потенційного позичальника, обумовленої суб'єктивною думкою фахівця.

Одним з варіантів розв’язання такої задачі є застосування алгоритмів, які вирішують задачі класифікації. Такого роду задачі з великим успіхом вирішуються за допомогою дерев рішень. Для демонстрації подібної технології скористаємося програмою Tree Analyzer з пакету Deductor. За вихідні дані була взята вибірка, що складається з 1000 записів, де кожен запис – це опис характеристик позичальника і параметр, що описує його поведінку під час погашення позики. При навчанні дерева використовувалися наступні чинники, що визначають позичальника: «N Паспорта», «ФІО», «Адреса», «Розмір позики», «Термін позики», «Мета позики» «Середньомісячний дохід», «Середньомісячна витрата» «Основний напрям витрат», «Наявність нерухомості» «Наявність автотранспорту», «Наявність банківського рахунку», «Наявність страховки», «Назва організації», «Галузева приналежність підприємства», «Термін роботи на даному підприємстві», «Напрям діяльності позичальника», «Термін роботи на даному напрямі», «Стать», «Забезпеченість позики», «Давати кредит» та інші. При цьому поля: «N Паспорта», «ФІО», «Адреса», «Назва організації» визначені алгоритмом вже до початку побудови дерева рішень як непридатні внаслідок практичної унікальності кожного із значень (рис. 4.38).

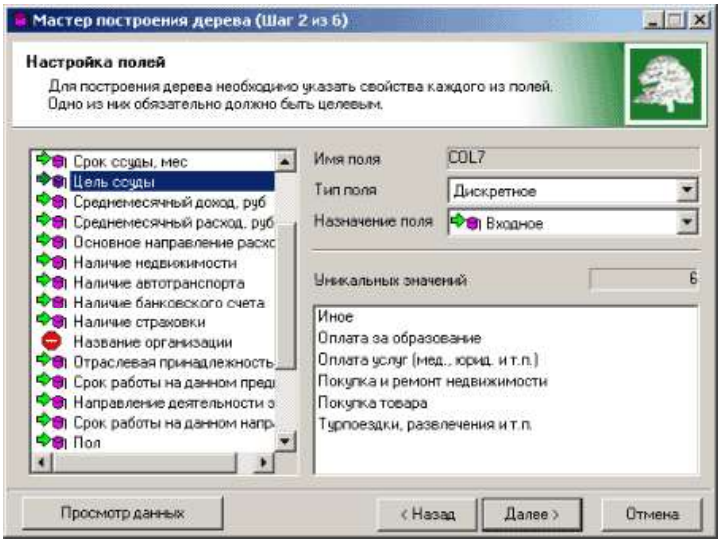


Рис. 4.38. Настройка визначальних і цільових чинників.

Цільовим полем є поле «Давати кредит», що набуває значень «Так» (True) і «Ні» (False). Ці значення можна інтерпретувати таким чином: «Ні» – платник або сильно прострочив з платежами, або не повернув частину грошей, «Так» – протилежність «Ні».

Після процесу побудови дерева рішень за допомогою програми Tree Analyzer отримуємо наступну модель оцінки кредитоспроможності фізичних осіб, що описує ситуацію, яка відноситься до певного банку. Ця модель представлена у вигляді ієрархічної структури правил – дерева рішень (рис. 4.39).

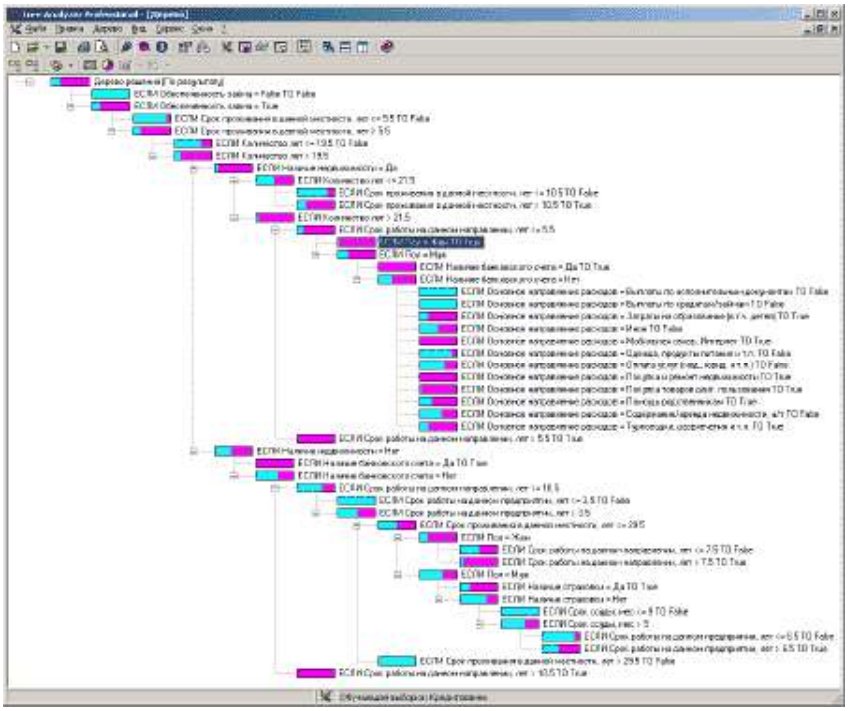


Рис. 4.39. Дерево рішень – модель визначення кредитоспроможності фізичних осіб.

Аналізуючи отримане дерево рішень, можна відмітити наступне:

1. За допомогою дерева рішень можна проводити аналіз значущих чинників. Таке можливе завдяки тому, що при визначенні параметра на кожному рівні ієрархії, по якому відбувається розділення на дочірні вузли, використовується критерій найбільшого усунення невизначеності. Таким чином, значиміші чинники, по яких проводиться класифікація, знаходяться на ближчій відстані

(глибині) від кореня дерева, чим менш значимі. Наприклад, чинник «Забезпеченість позики» більш значущий, ніж чинник «Термін мешкання в даній місцевості». А чинник «Основний напрям витрат» є значущим лише у поєднанні з іншими чинниками. Ще одним цікавим прикладом значущості різних чинників служить відсутність в побудованому дереві параметра «Наявність автотранспорту», що говорить про те, що на сьогоднішній день ця наявність не є визначальною при оцінці кредитоспроможності фізичної особи.

2. Можна відмітити, що такі показники як «Розмір позики», «Термін позики», «Середньомісячний дохід» і «Середньомісячна витрата» взагалі відсутні в отриманому дереві. Даний факт можна пояснити тим, що у вихідних даних присутній такий показник як «Забезпеченість позики», і оскільки цей чинник є точним узагальненням чотирьох вищеописаних показників, алгоритм побудови дерева рішень вибрав саме його.

Важливою особливістю побудованої моделі є те, що правила, по яких визначається приналежність позичальника до тієї або іншої групи, записані на природній мові. Наприклад, на основі побудованої моделі виходять наступні правила:

- ЯКЩО Забезпеченість позики = Так І Термін мешкання в даній місцевості, років > 5.5 І Кількість років > 19.5 І Наявність нерухомості = Так і Наявність банківського рахунку = Так ТО Давати кредит = Так (Достовірно на 98%)
- ЯКЩО Забезпеченість позики = Так І Термін мешкання в даній місцевості, років > 5.5 І Наявність нерухомості = Так і Кількість років > 21.5 І Термін роботи на даному напрямі, років <= 5.5 І Стать = Чол І Наявність банківського рахунку = Немає І Основний напрям витрат = Одяг, продукти харчування і тому подібне ТО Давати кредит = Ні (Достовірно на 88%).

На основі побудованої моделі також можна визначати приналежність потенційного позичальника до одного з класів. Для цього необхідно скористатися діалоговим вікном «Експеримент» програми Tree Analyzer, в

якому, послідовно відповідаючи на питання, можна отримати відповідь на питання: «Чи давати кредит» (рис. 4.40).

Рис. 4.40. Вікно «Експеримент».

Використовуючи такий підхід, можна усунути відразу обоє вищеописані недоліки скорингової системи оцінки кредитоспроможності.

Програмний комплекс Oracle Data Miner. Використання дерева рішень - це спосіб класифікації існуючих даних, визначення чинників або правил, які мають відношення до цільового результату (target result), і їх вживання для прогнозування результату, що означає:

1. Бізнес-користувачі можуть визначати чинники, які найбільшою мірою впливають на рішення про покупки.

2. Департаменти маркетингу можуть «цілитися» в «правильні» групи потенційних клієнтів, виключаючи тих, хто з малою вірогідністю купуватиме.

3. Аналітики даних і фінансові аналітики можуть прогнозувати продажу завдяки аналізу атрибутів потенційних клієнтів, про яких є дані.

4. Бізнес-аналітики можуть коректувати цілі і стратегії при змінах тенденцій

5. Компанії можуть реорганізовувати підтримку (support, enhancements, and desupport) для забезпечення максимального задоволення клієнтів.

Виробник пропонує два продукти, A і B. В цілому відгуки споживачів були позитивні, але власник підприємства-виробника хоче взнати, що можна змінити в підтримці для того щоб може підвищити рівень задоволення споживачів. У підприємства є вельми обмежена інформація про своїх споживачів, включаючи лише дані про продукт, який вони використовують, його версію і час здобуття його останньої модифікації. З використанням цієї інформації, отриманої від вибірки по клієнтській базі (sample customer population), і Oracle Data Miner це підприємство може створити модель дерева рішення (рис. 4.41).

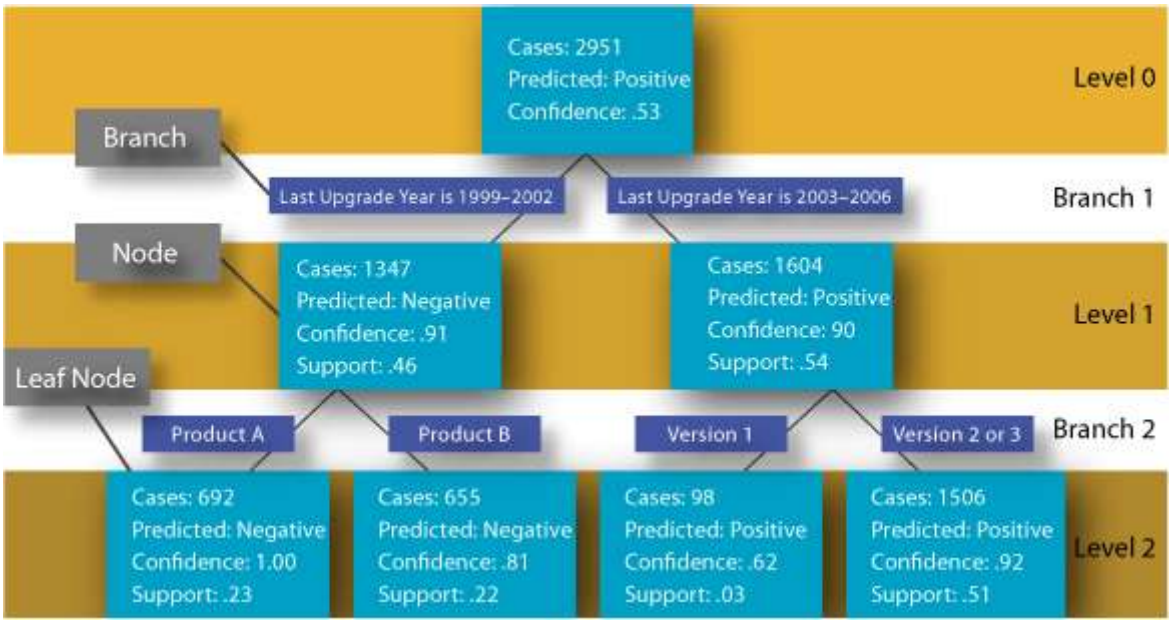


Рис. 4.41: Дерево рішень.

Кожен прямокутник в дереві називається вузлом (node) і кожна лінія називається віткою (branch) або ребром. Верхній прямокутник в дереві (або його корінь (root)) включає всі значення (all cases) цієї вибірки. Дерево рішень розділяє дані по атрибутах в спробі визначити кращих провісників (predictors) цільового значення (target value). Ці провісники формують правило (rule) або набір правил, які будучи застосовані до вузла, сформують результат. Ви можете думати про них, як про пропозиції IF-THEN для ухвалення рішень.

Oracle Data Miner аналізує всі атрибути в наборі даних. Якщо, наприклад, в даному наборі даних три атрибути, то Oracle Data Miner аналізує ці три атрибути. Якщо ж атрибутів 80, то Oracle Data Miner аналізує всі ці 80 атрибутів. Він визначає атрибут для першого розщеплювання (split) дерева рішення, яке щонайкраще ділить цільові дані (target data) на різні секції. З цим набором даних, розділеним надвоє, Oracle Data Miner може визначити атрибути для розщеплювань на наступному рівні. Зверніть увагу, що на рис. 4.41 Oracle Data Miner розщепнув гілки 2-го рівня по різних атрибутах. До останнього ряду вузлів посилаються як до термінального вузла або листа (terminal node, or leaf). З 4-х вузлів 2-го рівня один показує, що клієнти, які використовують версії 2 або 3 продукти A або B, до яких вони перейшли між 2003 і 2006, з 92% упевненості висловлюють позитивну думку про продукт. Інший же вузол показує, що клієнти, що використовують продукт A, до якого вони перейшли між 1999 і 2001, будуть, як передбачається, негативно відноситися до цього продукту. Безумовно, ця інформація буде корисної при планеруванні підтримки (рис. 4.42).

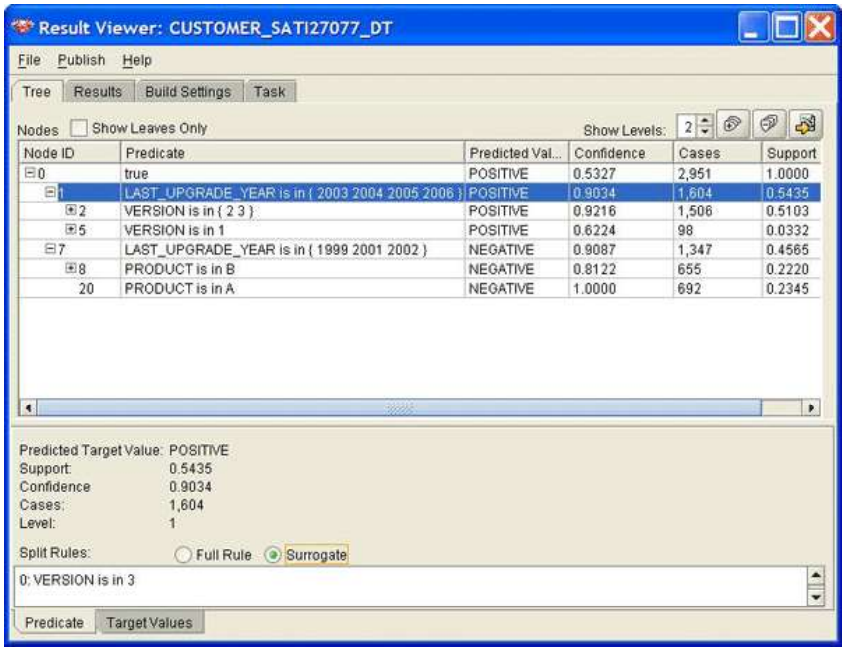


Рис. 4.42. Result Viewer показує правила розщеплювання.

З кожним листом асоціюється правило розщеплювання, яке з'являється внизу даного вікна, якщо клацнути по відповідному вузлу. Наприклад, останній лист (Node ID = 20 на рис. 4.42) передбачає негативну відповідь з 100% упевненістю. Oracle Data Miner дозволяє також побачити, які правила були використані для кожного запису. Натискуйте один із записів, а потім Rule button праворуч від результатів. Переглядач правив Rule Viewer з'явиться, як показано на рис. 4.43.

Рис 4.43. Переглядач правил Rule Viewer.

Система PolyAnalyst. У системі PolyAnalyst, реалізований алгоритм, заснований на критерії максимізації взаємної інформації (information gain). Тобто для розщеплювання вибирається незалежна змінна, що несе максимальну (у сенсі Шенона) інформацію про залежну змінну. Цей критерій на відміну від багатьох критеріїв, вживаних в інших системах Data Mining, має ясну інтерпретацію і дає розумні результати при найрізноманітніших статистичних параметрах даних, що вивчаються. Алгоритм DT є одним з найшвидших в PolyAnalyst (рис. 4.44).

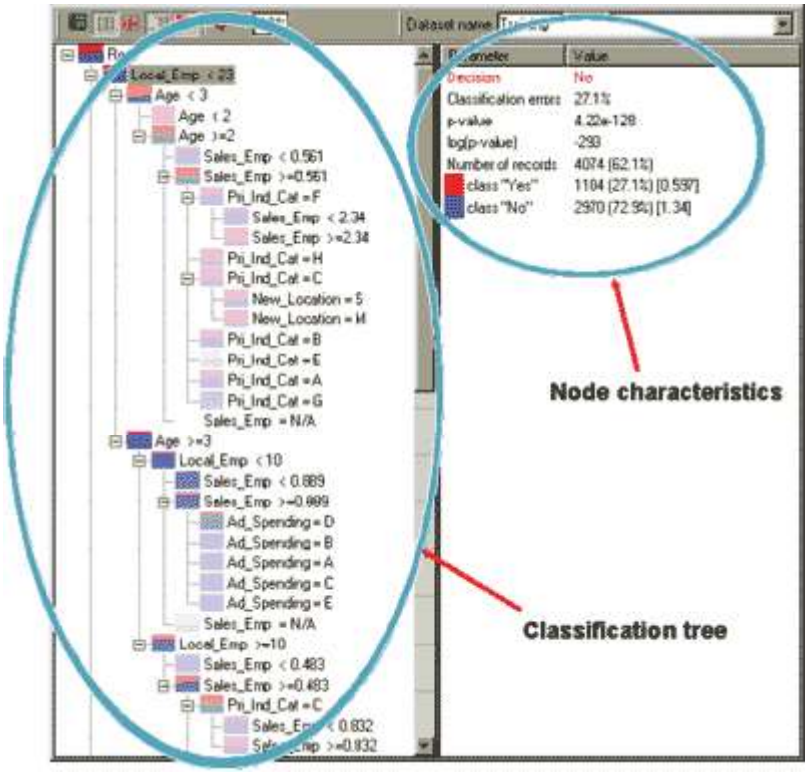
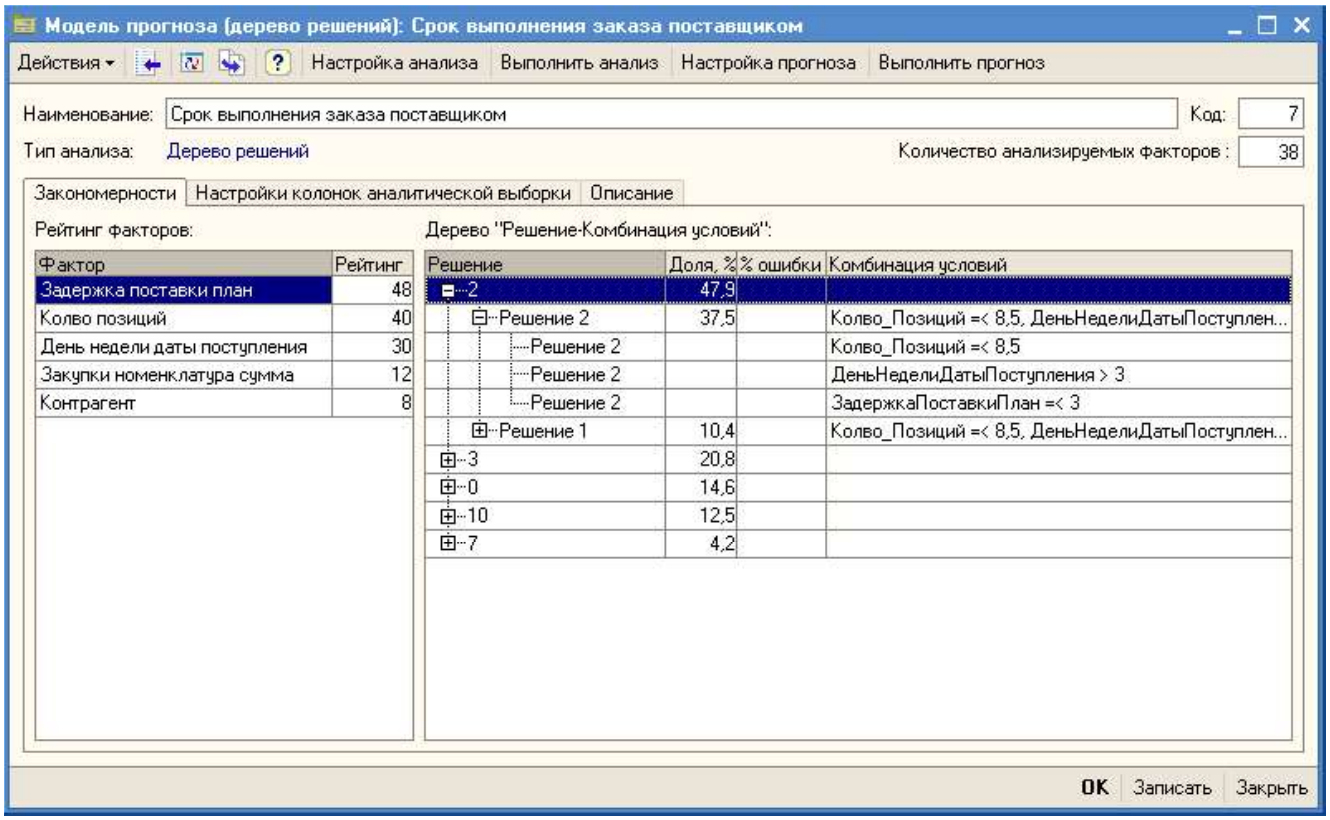


Рис. 4.44. Дерево рішень в системі PolyAnalyst.

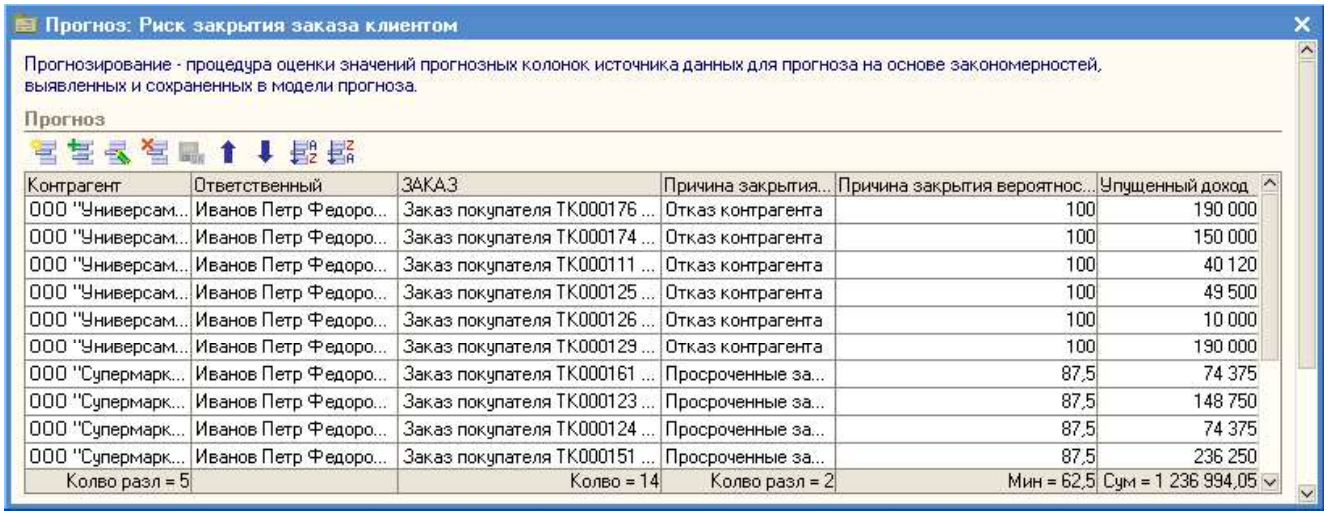
У разі, коли залежна змінна може приймати велику кількість різних значень, вживання методу дерев рішень стає неефективним. У такій ситуації в системі PolyAnalyst застосовується метод, званий лісом рішень (decision forest). При цьому будується сукупність дерев рішень - поодинці для кожного різного значення залежної змінної. Результатом прогнозу, заснованому на лісі рішень, є те значення залежної змінної, для якої відповідне дерево дає найбільш вірогідну оцінку.

Інформаційна система «1С: Підприємство 8.0». Даний алгоритм набув найбільшого поширення при виявленні причинно-наслідкових зв'язків в даних і описі поведінкових моделей. Типова зона застосовності дерев рішень – оцінка різних ризиків, наприклад, закриття замовлення клієнтом або його переходу до конкурента, невчасного постачання товару постачальником або прострочення оплати товарного кредиту. Як типові вхідні чинники моделі виступають сума і склад замовлення, поточне сальдо взаєморозрахунків, кредитний ліміт, відсоток передоплати, умови постачання і інші параметри, що характеризують об'єкт прогнозу. Адекватна оцінка ризику забезпечує ухвалення інформованих рішень

по оптимізації відношення доходность/ризик в діяльності компанії, а також корисна для збільшення реалістичності різних бюджетів (рис. 4.45).



а)



б)

Рис. 4.45. Вживання методу дерева рішень дозволяє на основі вхідних чинників моделі (а) отримувати оцінку ризиків ухвалення тих або інших управлінських рішень (б).

Як приклад, що ілюструє здатність цього алгоритму виявляти причинно-слідчі зв'язки, можна привести задачу оптимізації роботи відділу продажів. Для її вирішення як прогнозовану величину виберемо показник ефективності менеджерів по продажах, наприклад питому прибутковість на клієнта, а як чинники – сукупність даних, що потенційно впливають на результат. Алгоритм визначить чинники, що роблять найбільший вплив на результат, а також типові комбінації умов, що приводять до того або іншого результату. Більш того, підсистема «Аналіз даних» дозволить оцінити (спрогнозувати) очікувані значення цільового показника на підставі актуальних даних, а також провести прогноз «Що, якщо...», змінюючи показники, що подаються на вхід моделі. Результати аналізу і прогнозу за допомогою дерев рішень дозволяють істотно понизити вплив невизначеності бізнес-оточення на стан компанії, а також вирішити широкий спектр завдань, пов'язаних з виявленням складних і неочевидних причинно-наслідкових зв'язків.

Алгоритм «Дерево рішень» формує причинно-наслідкову ієрархію умов, що приводить до певних рішень. В результаті застосування цього методу до навчальної вибірки створюється ієрархічна (деревовидна) структура правил розщеплювання вигляду «Якщо... то...». Алгоритм аналізу (навчання моделі) зводиться до ітеративного процесу вичленення на кожному етапі найбільш значимих умов і переходів між ними. Умови можуть мати як кількісний, так і якісний характер і формують «гілки» цього абстрактного дерева. Його «листя» утворюють значення прогнозованого атрибуту (рішення), які також як і умови переходів можуть мати як якісне, таке і кількісне трактування. Сукупність цих умов, що накладаються на чинники і структура переходів між ними до кінцевого рішення і утворюють модель прогнозу.

Даний алгоритм набув найбільшого поширення при оцінці результатів різних подієвих ланцюжків і виявленні причинно-наслідкових зв'язків у вибірках. Управління значущістю і достовірністю моделі даного алгоритму здійснюється за допомогою параметрів «Тип спрощення», «Максимальна

глибина дерева» і «Мінімальна кількість елементів у вузлі». Як результат аналізу вибірки за допомогою алгоритму «Дерево рішень» виступають:

- рейтинг чинників, що є списком чинників, які зробили вплив на рішення, відсортований в порядку убування значущості («цитування» у вузлах дерева);
- зіставлення рішень (значень прогнозної колонки) і умов, що визначили їх, іншими словами - дерево «Наслідок-причина»;
- дерево «Причина-наслідок», що є сукупністю переходів між умовами, які визначають те або інше рішення (по суті - візуальне представлення моделі прогнозу).

В якості типового прикладу вживання інтелектуального аналізу даних стосовно конфігурації «1С: Управління торгівлею 8.0.» приведемо бізнес-сценарій – «Управління персоналом».

1. Сценарій – «Профілізація менеджерів відділу продажів за ключовими показниками ефективності». Визначення ефективності менеджерів (утримання, пошук клієнтів, ефективність комунікацій, інкасація умовної і безумовної дебіторської заборгованості, питомим показникам ефективності на клієнта і так далі) представляє інтерес не лише з точки зору формування системи матеріального стимулювання менеджерів, але і з точки зору ефективного нормування параметрів їх діяльності.

2. Алгоритм – «Дерева рішень».

3. Прогнозні атрибути - ключові показники ефективності відділу продажів (кількість ключових клієнтів, коефіцієнти відтоку і залучення, упущений дохід в місяць, притягнений дохід в місяць, дохід в місяць з клієнта, сумарні поступлення від клієнтів і так далі).

4. Основні чинники - кількість активних клієнтів, виручка, дохід, питомі показники на клієнта, ефективність комунікації. Залежно від прогнозних атрибутів склад чинників може істотно варіюватися.

5. Приклад закономірності. Менеджери, що забезпечують кращі показники інкасації дебіторської заборгованості мають коефіцієнт утримання > 0.8, коефіцієнт залучення > 0.25, кількість одночасно відкритих операцій не

більше 15, але не менше 10, інтенсивність подій в день не більше 10, але не менше 3, кількість активних клієнтів в періоді не менше 50, але не більше 100.

Автоматична класифікація тексту. Внутрішні вузли є термами, гілки, що відходять від них, характеризують вагу терма в аналізованому документі, а листя - категорії. Такий класифікатор категоризує випробовуваний документ, рекурсивно перевіряючи ваги вектора ознак по відношенню до порогів, виставлених для кожної з ваг, поки не досягне листа дерева (категорії). До цієї категорії (листа якого досяг класифікатор) і приписується аналізований документ.

На сьогоднішній день розроблена чимала кількість алгоритмів, що використовують для класифікації текстів технологією дерева рішень. Серед цих алгоритмів такі як CART, NewId, ITrule, CHAID і так далі Але найбільшою популярністю користується ID3 (Iterative Dichotomizer) і його розширені версії: загальнодоступна C4.5 і комерційна C5, які додають до алгоритму ID3 нові можливості. У основі багатьох сучасних алгоритмів лежить стратегія розділення і захвату (divide and conquer). Хай нам задана деяка навчальна множина, що містить отрубрицірованні документи, кожен з яких характеризується m атрибутами, причому один з них вказує на приналежність об'єкту до певного класу, та позначені класи (значення мітки класу). Тоді алгоритмом проводяться наступні перевірки:

- чи мають всі навчальні приклади одні і ті ж мітки класу (в цьому випадку дерево рішень для цього випадку є лист, що позначає клас);
- якщо немає, вибирають терм, який розділяє множину на класи документів, що мають ті ж самі значення вибраного параметра, і поміщає кожен такий клас в окреме піддерево;
- процес рекурсивно повторюється на піддеревах до тих пір, поки кожен лист дерева, не міститиме навчальні приклади, приписані до однієї з категорій.

Найважливіший крок - вибір значення параметра, який управляє вибором гілки. Вибраний параметр повинен розбити множину так, щоб отримувані у результаті підмножини складалися з документів, що належать до

одного класу, або мали мінімальне число «викидів» (тобто документів, що не відносяться до основного класу). Зазвичай, як такі параметри використовують приріст інформації (information gain) або критерій ентропії (entropy criterion). Часто результатом роботи алгоритмів побудови дерев рішень є складні дерева, що мають безліч вузлів і гілок. Такі дерева володіють правилами побудови класифікатора, що важко інтерпретуються. Окрім цього дерево, яке має величезну кількістю вузлів, розбиває навчальну множину на велике число підмножин, що складаються з малого числа документів. Для вирішення цієї проблеми зазвичай використовується метод, що отримав назву обрізка гілок (pruning). При використанні цього підходу, обрізання гілок, які не приводять до зростання помилки класифікації, відбувається від низу до верху. Класифікатор рухається з листя дерева, позначаючи вузли як листя, або замінюючи їх піддеревом. Не дивлячись на недоліки, переваги дерев рішень дозволили їм стати одними з найпопулярніших методів автоматичної класифікації тексту.

Машинне навчання. Мета методів машинного навчання – здобуття простих класифікуючих виразів, які були б легко зрозумілі для людини. Достоїнством таких методів є те, що під час роботи того або іншого методу не потрібна участь людини. У дослідженні, проведеному в рамках європейського проекту StatLog, був проведений аналіз статистичних методів (аналіз дискримінанта, кластер-аналіз і так далі), дерев рішень(C4.5, AC2, CART, NEWID, CN2, ITrule) і нейронних мереж (багатошарові мережі, РБФ-мережі, карти Кохонена) для вирішення задач класифікації. Дані були взяті з різних предметних областей: розпізнавання образів (рукописного тексту, автомобілів), медична діагностика (діабет, травми голови, сердечні захворювання), молекулярної біології (розпізнавання структури ДНК), видача кредитів і так далі В ході дослідження з'ясувалося, що дерева рішень показали найкращі результати у вирішенні наступних задач:

- оцінка кредитоспроможності кандидата на здобуття кредиту;
- діагностика несправностей в технічних системах;

- розміщення радіаторів в Space Shuttle.

Достоїнствами дерев рішень є:

1. На навчання дерев рішень потрібний значно менше часу, чим, наприклад, на навчання нейронних мереж. Результат роботи представляється у вигляді, що легко інтерпретується для людини. Класифікаційна модель, представлена у вигляді дерева є інтуїтивно зрозумілою для людини, на відміну від нейронних мереж, що є за своєю природою чорним ящиком.

2. На вхід алгоритму дерев рішень можна подавати будь-яку кількість параметрів, алгоритм сам вибере найбільш значимі параметри і лише вони фігуруватимуть в побудованому дереві. Це позбавляє користувача від необхідності визначати вхідні параметри. Знову ж таки, при використанні нейронних мереж ми повинні дуже обережно підходити до питання про вхідні поля, так, із зростанням кількості вхідних полів, збільшується час що витрачається на процес навчання, який і так є дуже довгим і викликає багато нарікань.

3. Точність прогнозу дерев рішень співпідставна з іншими методами побудови класифікаційних моделей (статистичні методи, нейроні мережі).

4. Існують масштабовані алгоритми дерев рішень SLIQ, SPRINT, тобто із зростанням числа прикладів час, що витрачається на навчання, зростає лінійно для побудови дерев рішень на надвеликих базах даних. Алгоритми побудови дерев рішень мають методи спеціальної обробки пропущених даних.

5. Класичні і сучасні методи статистики використовувані в задачах класифікації працюють лише з числовими даними, дерева рішень успішно працюють як з числовими, так і строковими значеннями. Крім того, деякі із статистичних методів є параметричними, тобто необхідно заздалегідь знати вигляд моделі або залежність між залежними і незалежними змінними. Наприклад, класифікатори, побудовані за принципом максимальної правдоподібності, передбачають, що дані мають нормальний розподіл. Вони також дозволяють витягувати правила на природній мові.

Самонавчальні системи прийняття рішень. Сьогодні актуальною проблемою є створення систем, що самостійно навчаються для роботи на фінансових ринках. Здібна до самонавчання система автоматичного управління (самоналагоджувальна система), в якій пристосування до умов, які випадково змінюються, забезпечується автоматичною зміною параметрів налаштування або шляхом автоматичного пошуку оптимального налаштування. У будь-якій іншій автоматичній системі управління є параметри, які впливають на стійкість і якості процесів управління і можуть бути змінені при регуляції (налаштуванні) системи. Якщо ці параметри залишаються незмінними, а умови функціонування (характеристики керованого об'єкту, збуджуючі дії) істотно змінюються, то процес управління може погіршати або навіть стати нестійким. Ручне налаштування системи часто виявляється складним, а інколи і неможливим. Використання в таких випадках самонавчальної системи технічно і економічно доцільно і навіть може виявитися єдиним способом надійного управління. У тих випадках, коли самоналаштування застосовується в системах управління в наслідок невірогідності знання властивостей об'єкту, система самооптимізації нагадує процес самонавчання. Система при цьому шляхом автоматичного пошуку немов би сама пізнає невідомі властивості керованого об'єкту і вчиться управляти цим об'єктом щонайкраще. Вживання таких систем вигідне в тих випадках, коли зовнішня дія може бути виміряна з метою аналізу його властивостей і коли зміна його форми є вирішальною для якостей роботи системи.

Особливість системи полягає в тому, що вона з часом набудовуватиметься, навчатиметься і нагромаджуватиме досвід користувача. Актуальність самонавчальної системи саме у тому, що людина не втручається в налаштування і її навчання. У системі, яка була розроблена, експертна оцінка буде індивідуальною для кожного інвестора, його набору чинників і отриманих трендів.

Побудова моделі прийняття рішення зводиться до рішення задачі класифікації. Для вирішення такої задачі з невизначеними початковими даними

на практиці найбільшого поширення набув метод дерева рішень, який дозволяє визначити набір правил класифікації типа «Якщо - То». Ці правила і будуть порадою системи користувачеві.

Синтез дерева рішень зводиться до такої задачі: відома вибірка інвестора

$$T = \{X^i, Y^i, i = \overline{1,n}\},$$

де X^i - вектор стану ринку, сформований інвестором у вигляді i оцінки; Y^i - реальний результат прийняття рішення для i оцінки інвестора; n - кількість оцінок. Необхідно за допомогою сформованої початкової вибірки побудувати функцію

$$X_1 \times X_2 \times ... \times X_n \to Y,$$

яка перетворює простір чинників на прогнозування класу зростання прибутку.

Автоматизована система, здібна до самонавчання і вироблення порад користувачеві складається з трьох блоків: база оцінок стану ринку, класифікація оцінки методом дерева рішень, автоматизована підсистема прийняття рішень (рис. 4.46).

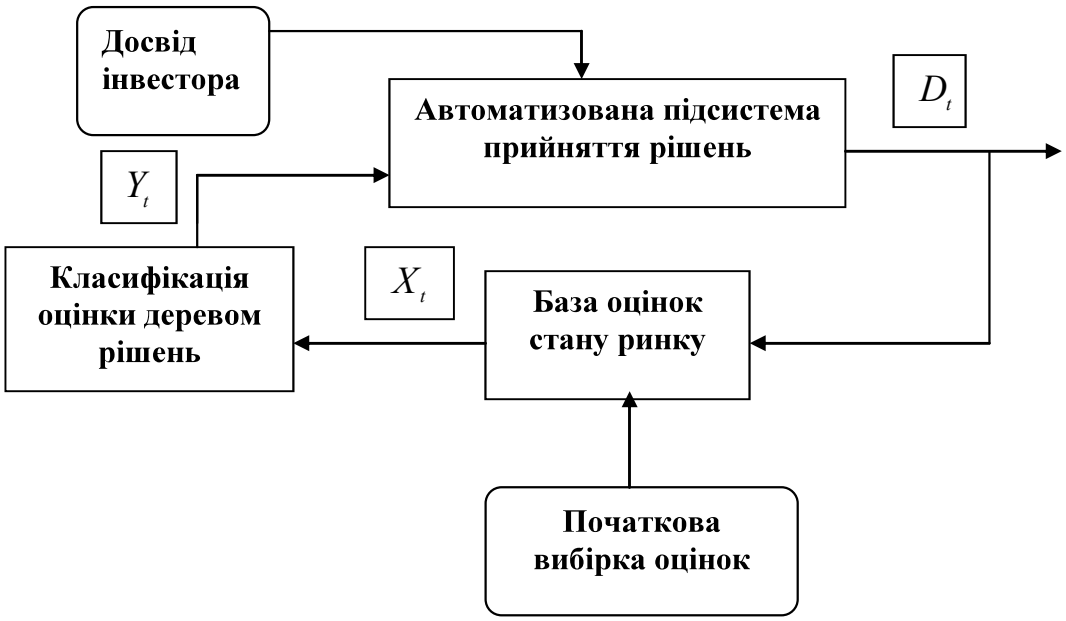


Рис. 4.46. Структурна схема автоматизованої системи прийняття рішень.

За допомогою початкової вибірки оцінок стану ринку система формує дерево рішень, яке надалі дозволить класифікувати подальші вектори оцінок у клас зростання прибутку. Отримавши результати класифікації, система виробляє пораду інвесторові, на основі якого і враховуючи власний досвід інвестор приймає рішення що заноситься в базу оцінок стану ринку і надалі використовуватиметься системою для класифікації наступної оцінки.

Тест.

1. В якому випадку, на Ваш погляд, виникають асоціації?

а) якщо виникає потреба аналізу великої кількості даних;

б) в випадку, коли треба зв'язати дані з різних баз даних;

в) якщо потрібно підвищити ефективність процес продажу товарів;

г) в випадку, коли декілька подій зв'язано одна з одною.

2. Під асоціативними правилами Ви розумієте …

а) кількісний аналіз подій;

б) опис зв'язків між декількома подіями;

в) якісний аналіз подій.

3. Цікавими вважаються такі асоціативні правила, які …

а) перевищують певний поріг;

б) мають кількісні та якісні характеристики подій;

в) пов'язують найбільшу кількість подій.

4. Дайте формальне визначення поняттю «асоціативне правило»:

а) асоціативним правилом називається імплікація $X \Rightarrow Y$, де $X \subset I$, $Y \subset I$ і $X \cap Y = 0$;

б) асоціативним правилом називається імплікація $X \Rightarrow Y$, де $X \not\subset I$, $Y \not\subset I$ і $X \cap Y = 0$;