Міністерство освіти і науки України

Черкаський державний технологічний університет

Кафедра програмного забезпечення автоматизованих систем

**ЗВІТ**

з лабораторної роботи №5

з предмету «Аналіз великих даних»

|  |  |
| --- | --- |
| Перевірив:  к. т. н., д. каф. ПЗАС  Рідкокаша А.А.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2019 р. | Виконав:  студент 1-го курсу  групи МПЗ-1904  Гаврилюк В. Є. |

Черкаси 2019

**Лабораторна робота №5**

**Тема:** Побудова й використання асоціативних правил.

**Мета:** Побудувати асоціативні правила на основі фрагменту бази даних транзакцій, використовуючи алгоритм Apriori.

**Завдання:**

1. За наведеною у початкових даних таблицею, яка є фрагментом бази даних транзакцій побудувати асоціативні правила, використовуючи алгоритм Apriori. В ході побудови застосовувати наступні параметри: мінімальна підтримка правила 20%, мінімальна достовірність правила 80%.
2. Розрахувати додаткові характеристики асоціативних правил. Зробити висновки щодо значущості кожного з отриманих правил.
3. Варіюючи в широких межах величини підтримки й достовірності знайти оптимальні граничні значення цих величин для розглянутої предметної області. Результат пояснити.

**Короткі теоретичні відомості**

Сучасні бази даних мають дуже великі розміри, що досягають гіга- і терабайтів, та тенденцію до подальшого збільшення. І тому, для знаходження асоціативних правил, потрібні ефективні масштабовані алгоритми, що дозволяють вирішити задачу за прийнятний час.

Для того, щоб було можливо застосувати алгоритм, необхідно провести попередню обробку даних: по-перше, привести всі дані до бінарним виду; по-друге, змінити структуру даних.

|  |  |  |
| --- | --- | --- |
| Номер транзакції | Найменування елемента | Кількість |
| 1001 | А | 2 |
| 1001 | D | 3 |
| 1001 | E | 1 |
| 1002 | А | 2 |
| 1002 | F | 1 |
| 1003 | B | 2 |
| 1003 | A | 2 |
| 1003 | C | 2 |

*Таблиця 1 - Звичайний вигляд бази даних транзакцій.*

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |

*Таблиця 2 - Нормалізований вид.*

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 в іншому випадку. Зауважимо, що початковий вигляд таблиці може бути не описаними в таблиці 1. Головне, щоб дані були перетворені до нормалізовано виду, інакше алгоритм неможливо буде застосувати.

Більш того, як видно з таблиці, всі елементи впорядковані в алфавітному порядку (якщо це числа, вони повинні бути впорядковані в числовому порядку).

Такі алгоритми працюють в два етапи, не є винятком і аналізований нами алгоритм Apriori. На першому кроці необхідно знайти набори елементів, які часто зустрічаються, а потім, на другому, витягти з них правила. Кількість елементів в наборі будемо називати розміром набору, а набір, що складається з k елементів, - k-елементним набором.

**Властивість анти-монотонності**

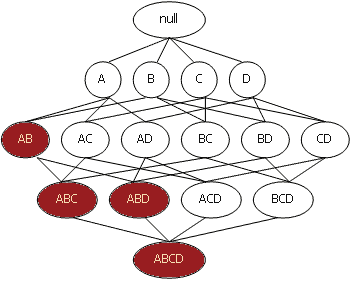
Виявлення наборів елементів, які часто зустрічаються - операція, яка потребує багато обчислювальних ресурсів і, відповідно, часу. Примітивний підхід до вирішення даного завдання - простий перебір всіх можливих наборів елементів. Це потребує O (2 | I |) операцій, де | I | - кількість елементів. Apriori використовує одну з властивостей підтримки, з якого випливає: підтримка будь-якого набору елементів не може перевищувати мінімальної підтримки будь-якого з його підмножин. Наприклад, підтримка 3-елементного набору {Хліб, Масло, Молоко} буде завжди менше або дорівнюватиме підтримці 2-елементних наборів {Хліб, Масло}, {Хліб, Молоко}, {Масло, Молоко}. Справа в тому, що будь-яка транзакція, яка містить {Хліб, Масло, Молоко}, також повинна містити {Хліб, Масло}, {Хліб, Молоко}, {Масло, Молоко}, причому зворотне не вірно.

Це властивість носить назву анти-монотонності і служить для зниження розмірності простору пошуку. Не май ми в наявності такої властивості, знаходження багатоелементних наборів було б практично нездійсненним завданням в зв'язку з ростом обчислень.

Властивості анти-монотонності можна дати і інше формулювання: з ростом розміру набору елементів підтримка зменшується, або залишається такою ж. З усього вищесказаного випливає, що будь-який k-елементний набір буде часто зустрічатися тоді і тільки тоді, коли всі його (k-1)-елементні підмножини будуть часто зустрічатися.

Всі можливі набори елементів з I можна представити у вигляді решітки, що починається з порожньої множини, потім на 1 рівні 1-елементні набори, на 2-му - 2-елементні і т.д. На k рівні представлені k-елементні набори, пов'язані з усіма своїми (k-1)-елементними підмножинами.

Розглянемо малюнок 1, який ілюструє набір елементів I - {A, B, C, D}. Припустимо, що набір з елементів {A, B} має підтримку нижче заданого порогу і, відповідно, не є таким, який часто зустрічається. Тоді, відповідно до властивості анти-монотонності, всі його супермножини також не є такими, які часто зустрічаються, і відкидаються. Вся ця гілка, починаючи з {A, B}, виділена фоном. Використання цієї евристики дозволяє істотно скоротити простір пошуку.



*Малюнок 1 - Набір елементів I - {A, B, C, D}.*

**Алгоритм Apriori**

На першому кроці алгоритму підраховуються 1-елементні набори, які часто зустрічаються. Для цього необхідно пройтись по всьому набору даних і підрахувати для них підтримку, тобто скільки разів зустрічається в базі.

Наступні кроки будуть складатися з двох частин: генерації наборів елементів, які потенційно часто зустрічаються (їх називають кандидатами) і підрахунку підтримки для кандидатів.

Описаний вище алгоритм можна записати у вигляді наступного псевдо-коду:

F1 = {часто зустрічаються 1-елементні набори}

для (k = 2; Fk-1 <> ⊘; k ++) {

Ck = Apriorigen (Fk-1) // генерація кандидатів

для всіх транзакцій t ε T {

Ct = subset (Ck, t) // видалення надлишкових правил

для всіх кандидатів c ε Ct

c.count ++

}

Fk = {c ε Ck | c.count> = minsupport} // відбір кандидатів

}

Результат ∪ Fk

Наведемо функцію генерації кандидатів. На цей раз немає ніякої необхідності знову звертатися до бази даних. Для того, щоб отримати k-елементні набори, скористаємося (k-1)-елементними наборами, які були визначені на попередньому етапі і є такими, що часто зустрічаються.

Згадаймо, що наш початковий набір зберігається у впорядкованому вигляді. Генерація кандидатів також буде складатися з двох кроків.

1. Об'єднання. Кожен кандидат Ck буде формуватися шляхом розширення набору, що часто зустрічається, розміру (k-1) додаванням елемента з іншого (k-1)-елементного набору. Наведемо алгоритм цієї функції Apriorigen у вигляді невеликого SQL-подібного запиту.

insert into Ck

select p.item1, p.item2, ..., p.itemk-1, q.itemk-1

From Fk-1 p, Fk-1 q

where p.item1 = q.item1, p.item2 = q.item2, ..., p.itemk-2 = q.itemk-2, p.itemk-1 <q.itemk-1

1. Видалення надлишкових правил. На підставі властивості анти-монотонності, слід видалити всі набори c ε Ck якщо хоча б одне з його (k-1) підмножин не є таким, що часто зустрічається.

Після генерації кандидатів наступним завданням є підрахунок підтримки для кожного кандидата. Очевидно, що кількість кандидатів може бути дуже великою і потрібен ефективний спосіб підрахунку. Самий тривіальний спосіб - порівняти кожну транзакцію з кожним кандидатом. Але це далеко не найкраще рішення. Набагато швидше і ефективніше використовувати підхід, заснований на зберіганні кандидатів в хеш-дереві. Внутрішні вузли дерева містять хеш-таблиці з покажчиками на нащадків, а листя - на кандидатів. Це дерево нам стане в нагоді для швидкого підрахунку підтримки для кандидатів.

Хеш-дерево будується кожен раз, коли формуються кандидати. Спочатку дерево складається тільки з кореня, який є листом, і не містить ніяких кандидатів-наборів. Кожен раз коли формується новий кандидат, він заноситься в корінь дерева і так до тих пір, поки кількість кандидатів в корені-листі не перевищить певного порогу. Як тільки кількість кандидатів стає більше порога, корінь перетворюється в хеш-таблицю, тобто стає внутрішнім вузлом, і для нього створюються нащадки-листя. І всі приклади розподіляються по вузлах-нащадкам згідно хеш-значень елементів, що входять в набір, і т.д. Кожен новий кандидат хешується на внутрішніх вузлах, поки він не досягне першого вузла-листа, де він і буде зберігатися, поки кількість наборів знову ж таки не перевищить поріг.

Хеш-дерево з кандидатами-наборами побудовано, тепер, використовуючи хеш-дерево, 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 Ck ∩ Ti = Ck. На кореневому рівні хеш-функція застосовується до кожного елементу з транзакції. Далі, на другому рівні, хеш-функція застосовується до других елементів і т.д. На k-рівні хешується 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 мінімального порогу. Кандидати, для яких ця умова виконується, переносяться в розряд таких, що часто зустрічаються. Крім того, слід запам'ятати і підтримку набору, вона нам стане в нагоді при добуванні правил. Ці ж дії застосовуються для знаходження (k+1)-елементних наборів і т.д.

Після того як знайдені всі набори елементів, що часто зустрічаються, можна приступити безпосередньо до генерації правил.

Витяг правил - менш трудомістке завдання. По-перше, для підрахунку достовірності правила досить знати підтримку самого набору і множини, що лежить в умови правила. Наприклад, є набір {A, B, C}, що часто зустрічається, і потрібно підрахувати достовірність для правила AB ⇒ C. Підтримка самого набору нам відома, а й його множини {A, B}, що лежить в умови правила, також є часто зустрічається в силу властивості анти-монотонності, і значить його підтримка нам відома. Тоді ми легко зможемо підрахувати достовірність. Це позбавляє нас від небажаного перегляду бази транзакцій, який потрібен був в тому випадку якби ця підтримка була невідома.

Щоб витягти правило з набору F, який часто зустрічається, слід знайти всі його непусті підмножини. І для кожної підмножини s ми зможемо сформулювати правило s ⇒ (F - s), якщо достовірність правила conf (s ⇒ (F - s)) = supp (F) / supp (s) не гірша від порогу minconf.

Зауважимо, що чисельник залишається постійним. Тоді достовірність має мінімальне значення, якщо знаменник має максимальне значення, а це відбувається в тому випадку, коли в умови правила є набір, що складається з одного елемента. Всі супермножини даної множини мають меншу або рівну підтримку і, відповідно, більше значення достовірності. Ця властивість може бути використано при добуванні правил. Якщо ми почнемо витягувати правила, розглядаючи спочатку тільки один елемент в умові правила, і це правило має необхідну підтримку, тоді всі правила, де в умові стоять супермножини цього елемента, також мають значення достовірності вище заданого порогу. Наприклад, якщо правило A ⇒ BCDE задовольняє мінімальний поріг достовірності minconf, тоді AB ⇒ CDE також задовольняє. Для того, щоб витягти всі правила використовується рекурсивна процедура. Важливе зауваження: будь-яке правило, складене з набору, що часто зустрічається, має містити всі елементи набору. Наприклад, якщо набір складається з елементів {A, B, C}, то правило A ⇒ B не повинно розглядатися.

**Хід роботи**

|  |  |
| --- | --- |
| **№ транзакції** | **Замовлення відвідувачів** |
| 1 | Гаспачо, різотто, шампанське |
| 2 | Телячий стейк, фруктовий суп, шампанське |
| 3 | Шампанське, грецький салат, карпачо, фруктовий суп |
| 4 | Різотто, шампанське, грецький салат, карпачо |
| 5 | Карпачо, телячий стейк, гаспачо |
| 6 | Фруктовий суп, телячий стейк, карпачо, грецький салат |
| 7 | Грецький салат, шампанське |
| 8 | Гаспачо, грецький салат, різотто |
| 9 | Фруктовий суп, телячий стейк, карпачо |
| 10 | Карпачо, шампанське |
| 11 | Різотто, гаспачо, карпачо, фруктовий суп |
| 12 | Телячий стейк, карпачо, фруктовий суп |
| 13 | Фруктовий суп, шампанське, телячий стейк, карпачо |
| 14 | Шампанське, різотто, грецький салат, карпачо, гаспачо |

*Таблиця 3 - Транзакції в ресторані середземноморської кухні.*

Користуючись вищенаведеним алгоритмом, насамперед перевіряємо, чи всі замовлення відповідають умовам “частих наборів”, тобто перевіряємо підтримку наборів першого рівня. Заданий рівень підтримки 20% свідчить, що страва має зустрічатися в транзакціях не менше 3 разів .

Для перевірки представимо транзакції у бінарній формі й підрахуємо кількість замосленнь кожної страви (див. табл. 3).

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **№ транзакції** | **Телячий стейк** | **Карпачо** | **Гаспачо** | **Шампанське** | **Різотто** | **Фруктовий суп** | **Грецький салат** |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 2 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 3 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| 4 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 5 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 6 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| 7 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 8 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| 9 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| 10 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 11 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 12 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| 13 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| 14 | 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| **Сума** | **6** | **10** | **5** | **8** | **5** | **7** | **6** |

*Таблиця 4 - Нормалізований вид множини транзакцій.*

Як видно з таблиці 4, всі без винятку набори першого рівня (окремі страви) є частими. Переходимо до другого рівня, генеруючи всі можливі пари страв. Самі набори другого рівня та їх підтримка наведені у таблиці 5.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Набір** | **Кількість** | **Набір** | **Кількість** | **Набір** | **Кількість** |
| Телячий стейк, карпачо | 5 | Карпачо, шампанське | 5 | Гаспачо, грецький салат | 2 |
| Телячий стейк, гаспачо | 1 | Карпачо, різотто | 3 | Шампанське, різотто | 3 |
| Телячий стейк, шампанське | 2 | Карпачо, фруктовий суп | 6 | Шампанське, фруктовий суп | 3 |
| Телячий стейк, різотто | 0 | Карпачо, грецький салат | 4 | Шампанське, грецький салат | 4 |
| Телячий стейк, фруктовий суп | 5 | Гаспачо, шампанське | 2 | Різотто, фруктовий суп | 1 |
| Телячий стейк, грецький салат | 1 | Гаспачо, різотто | 4 | Різотто, грецький салат | 3 |
| Карпачо, гаспачо | 3 | Гаспачо, фруктовий суп | 1 | Фруктовий суп, грецький салат | 2 |

*Таблиця 5 - Предметні набори з двох овочів.*

Як видно з таблиці 5, за прийнятими критеріями підтримки частими є лише 12 наборів з двох страв:

{телячий стейк, карпачо}; {телячий стейк, фруктовий суп}; {карпачо, гаспачо}; {карпачо, шампанське}; {карпачо, різотто}; {карпачо, фруктовий суп}; {карпачо, грецький салат}; {гаспачо, різотто}; {шампанське, різотто}; {шампанське, фруктовий суп}; {шампанське, грецький салат}; {різотто, грецький салат}.

Далі використовуємо отримані набори з двох страв для генерації множин з трьох предметів . Для цього знаходимо всі об’єднання множин рівня , що мають спільний перший елемент. Наприклад, {телячий стейк, карпачо} + {телячий стейк, фруктовий суп} = {телячий стейк, карпачо, фруктовий суп}.

Утворені набори з трьох страв та їх частоти представимо у вигляді таблиці.

|  |  |  |  |
| --- | --- | --- | --- |
| **Набір** | **Кількість** | **Набір** | **Кількість** |
| Телячий стейк, карпачо, фруктовий суп | 4 | Карпачо, шампанське, грецький салат | 3 |
| Карпачо, гаспачо, шампанське | 1 | Карпачо, різотто, фруктовий суп | 1 |
| Карпачо, гаспачо, різотто | 2 | Карпачо, різотто, грецький салат | 2 |
| Карпачо, гаспачо, фруктовий суп | 1 | Карпачо, фруктовий суп, грецький салат | 2 |
| Карпачо, гаспачо, грецький салат | 1 | Шампанське, різотто, фруктовий суп | 0 |
| Карпачо, шампанське, різотто | 2 | Шампанське, різотто, грецький салат | 2 |
| Карпачо, шампанське, фруктовий суп | 2 | Шампанське, фруктовий суп, грецький салат | 1 |

*Таблиця 6 - Предметні набори з трьох страв.*

З огляду на отримані дані, множин третього порядку всього дві:

{телячий стейк, карпачо, фруктовий суп}; {карпачо, шампанське, грецький салат}.

Подальше виділення чистих наборів сенсу не має. Таким чином, в розглянутій задачі виділено 14 однопредметних частих наборів, 12 – двопредметних та 2 – трьопредметних. Маючи їх всі, переходимо до генерації асоціативних правил.

Процедура генерації правил виконується послідовно для частих наборів усіх рівнів, починаючи з вищого і передбачає:

1. Виділення в наборі https://studfiles.net/html/2706/830/html_8yIz8RCgIl.eG2H/img-w6vttu.png можливого часткового набору ;
2. Якщо , то записується асоціація вигляду , де , або точніше ;
3. Всі отримані правила оцінюються за підтримкою та достовірністю. Такі, що не задовольняють обраним критеріям, відкидаються.

Проілюструємо процедуру наступним прикладом. Набір {телячий стейк, карпачо, фруктовий суп} може бути розділений на наступні не порожні часткові набори: {телячий стейк}, {карпачо}, {фруктовий суп}, {телячий стейк, карпачо}, {телячий стейк, фруктовий суп}, {карпачо, фруктовий суп}. Відповідно може бути записано 12 варіантів правил:

*Якщо {телячий стейк}, то {карпачо, фруктовий суп};*

*Якщо {карпачо}, то {телячий стейк, фруктовий суп};*

*Якщо {фруктовий суп}, то {телячий стейк, карпачо};*

*Якщо {телячий стейк, карпачо}, то {фруктовий суп};*

*Якщо {телячий стейк, фруктовий суп}, то {карпачо};*

*Якщо {карпачо, фруктовий суп}, то {телячий стейк}.*

*Якщо {карпачо}, то {шампанське, грецький салат};*

*Якщо {шампанське}, то {карпачо, грецький салат};*

*Якщо {грецький салат}, то {карпачо, шампанське};*

*Якщо {карпачо, шампанське}, то {грецький салат};*

*Якщо {карпачо, грецький салат}, то {шампанське};*

*Якщо {шампанське, грецький салат}, то {карпачо}.*

Для першого правила підтримка – це відсоток рядків початкової бази даних, де одночасно зустрічаються як умова *{телячий стейк},*так і наслідок *{карпачо, фруктовий суп}.*Таких рядків 4 з 14 транзакцій. Тобто . Достовірність цього правила вимірюється як відношення частоти зустрічі усього набору до частоти умови, тобто .

Аналогічно розраховуємо показники значущості для всіх інших можливих правил третього рівня та заносимо результат в таблицю 7.

|  |  |  |
| --- | --- | --- |
| **Якщо умова, то наслідок** | **Підтримка** | **Достовірність** |
| Якщо {телячий стейк}, то {карпачо й фруктовий суп} | 4 / 14 = 28,6% | 4 / 6 = |
| Якщо {карпачо}, то {телячий стейк й фруктовий суп} | 4 / 14 = 28,6% | 4 / 10 = 40% |
| Якщо {фруктовий суп}, то {телячий стейк й карпачо} | 4 / 14 = 28,6% | 4 / 7 = 57,1% |
| Якщо {телячий стейк й карпачо}, то {фруктовий суп} | 4 / 14 = 28,6% | 4 / 5 = 80% |
| Якщо {телячий стейк й фруктовий суп}, то {карпачо} | 4 / 14 = 28,6% | 4 / 5 = 80% |
| Якщо {карпачо й фруктовий суп}, то {телячий стейк} | 4 / 14 = 28,6% | 4 / 6 = 66,7% |
| Якщо {карпачо}, то {шампанське, грецький салат} | 3 / 14 = 21,4% | 3 / 10 = 30% |
| Якщо {шампанське}, то {карпачо, грецький салат} | 3 / 14 = 21,4% | 3 / 8 = 37,5% |
| Якщо {грецький салат}, то {карпачо, шампанське} | 3 / 14 = 21,4% | 3 / 6 = 50% |
| Якщо {карпачо, шампанське}, то {грецький салат} | 3 / 14 = 21,4% | 3 / 5 = 60% |
| Якщо {карпачо, грецький салат}, то {шампанське} | 3 / 14 = 21,4% | 3 / 4 = 75% |
| Якщо {шампанське, грецький салат}, то {карпачо} | 3 / 14 = 21,4% | 3 / 4 = 75% |

*Таблиця 7 - Асоціативні правила із двома предметами в умові.*

За викладеною вище методикою розраховуємо показники значущості для всіх інших можливих правил третього рівня та заносимо результат в таблицю.

|  |  |  |
| --- | --- | --- |
| **Якщо умова, то наслідок** | **Підтримка** | **Достовірність** |
| Якщо {телячий стейк}, то {карпачо} | 5 / 14 = 35,7% | 5 / 6 = 83,3% |
| Якщо {карпачо}, то {телячий стейк} | 5 / 14 = 35,7% | 5 / 10 = 50% |
| Якщо {телячий стейк}, то {фруктовий суп} | 5 / 14 = 35,7% | 5 / 6 = 83,3% |
| Якщо {фруктовий суп}, то {телячий стейк} | 5 / 14 = 35,7% | 5 / 7 = 71,4% |
| Якщо {карпачо}, то {гаспачо} | 3 / 14 = 21,4% | 3 / 10 = 30% |
| Якщо {гаспачо}, то {карпачо} | 3 / 14 = 21,4% | 3 / 5 = 60% |
| Якщо {карпачо}, то {шампанське} | 5 / 14 = 35,7% | 5 / 10 = 50% |
| Якщо {шампанське}, то {карпачо} | 5 / 14 = 35,7% | 5 / 8 = 62,5% |
| Якщо {карпачо}, то {різотто} | 3 / 14 = 21,4% | 3 / 10 = 30% |
| Якщо {різотто}, то {карпачо} | 3 / 14 = 21,4% | 3 / 5 = 60% |
| Якщо {карпачо}, то {фруктовий суп} | 6 / 14 = 42,9% | 6 / 10 = 60% |
| Якщо {фруктовий суп}, то {карпачо} | 6 / 14 = 42,9% | 6 / 7 = 85,7% |
| Якщо {карпачо}, то {грецький салат} | 4 / 14 = 28,6% | 4 / 10 = 40% |
| Якщо {грецький салат}, то {карпачо} | 4 / 14 = 28,6% | 4 / 6 = 66,7% |
| Якщо {гаспачо}, то {різотто} | 4 / 14 = 28,6% | 4 / 5 = 80% |
| Якщо {різотто}, то {гаспачо} | 4 / 14 = 28,6% | 4 / 5 = 80% |
| Якщо {шампанське}, то {різотто} | 3 / 14 = 21,4% | 3 / 8 = 37,5% |
| Якщо {різотто}, то {шампанське} | 3 / 14 = 21,4% | 3 / 5 = 60% |
| Якщо {шампанське}, то {фруктовий суп} | 3 / 14 = 21,4% | 3 / 8 = 37,5% |
| Якщо {фруктовий суп}, то {шампанське} | 3 / 14 = 21,4% | 3 / 7 = 42,9% |
| Якщо {шампанське}, то {грецький салат} | 4 / 14 = 28,6% | 4 / 8 = 50% |
| Якщо {грецький салат}, то {шампанське} | 4 / 14 = 28,6% | 4 / 6 = 66,7% |
| Якщо {різотто}, то {грецький салат} | 3 / 14 = 21,4% | 3 / 5 = 60% |
| Якщо {грецький салат}, то {різотто} | 3 / 14 = 21,4% | 3 / 6 = 50% |

*Таблиця 8 - Асоціативні правила з одним предметом в умові*

Повертаючись до початкових умов задачі (мінімальна підтримка 20%, мінімальна достовірність 80%), відкидаємо правила, які не задовольняють вимогам до значущості. Аби впорядкувати правила (встановити пріоритет їх перевірки при використанні у базі знань), розрахуємо синтетичний показник значущості, який дорівнює добутку підтримки на достовірність.

Результати роботи представимо у вигляді таблиці 9.

|  |  |  |  |
| --- | --- | --- | --- |
| **Якщо умова, то наслідок** | **Підтримка, S** | **Достовірність, C** | **C × S** |
| Якщо {гаспачо}, то {різотто} | 4 / 14 = 28,6% | 4 / 5 = 80% | 0,2288 |
| Якщо {різотто}, то {гаспачо} | 4 / 14 = 28,6% | 4 / 5 = 80% | 0,2288 |
| Якщо {телячий стейк й карпачо}, то {фруктовий суп} | 4 / 14 = 28,6% | 4 / 5 = 80% | 0,2288 |
| Якщо {телячий стейк й фруктовий суп}, то {карпачо} | 4 / 14 = 28,6% | 4 / 5 = 80% | 0,2288 |
| Якщо {телячий стейк}, то {карпачо} | 5 / 14 = 35,7% | 5 / 6 = 83,3% | 0,2972 |
| Якщо {телячий стейк}, то {фруктовий суп} | 5 / 14 = 35,7% | 5 / 6 = 83,3% | 0,2972 |
| Якщо {фруктовий суп}, то {карпачо} | 6 / 14 = 42,9% | 6 / 7 = 85,7% | 0,3677 |

*Таблиця 9 - Асоціативні правила після відсіювання за підтримкою та достовірністю*

Таким чином, у результаті застосування алгоритму Аргіогі нам удалося виявити 7 асоціативних правил, які з достовірністю не менше 80% показують, які страви з вихідного меню найчастіше замовляють разом. Це знання дозволить розробити більш досконалу маркетингову стратегію, оптимізувати пропозиції відвідувачам та розмістити страви в розділі меню із сетіми страв.

**Висновок:** На даній лабораторній роботі я побудував асоціативні правила на основі фрагменту бази даних транзакцій, використовуючи алгоритм Apriori. Ознайомився з алгоритмом Apriori та основними принципами його роботи. Виявив 7 асоціативних правил на основі обробки фрагменту бази даних ресторану середземноморської кухні, що дозволять розробити більш досконалу маркетингову систему.