VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

ZÍSKÁVÁNÍ VÍCEÚROVŇOVÝCH ASOCIAČNÍCH PRAVIDEL

MINING MULTIPLE LEVEL ASSOCIATION RULES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **THANH LAM NGUYENOVÁ**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. VLADIMÍR BARTÍK, Ph.D.**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2021/2022

Zadání bakalářské práce

Studentka: **Nguyenová Thanh Lam**
Program: Informační technologie
Název: **Získávání víceúrovňových asociačních pravidel**
Mining Multiple Level Association Rules
Kategorie: Data mining

Zadání:

1. Prostudujte problematiku získávání znalostí z databází, podrobněji se zaměřte na víceúrovňová asociační pravidla.
2. Po dohodě s vedoucím zvolte několik metod, tyto metody detailně prostudujte.
3. Navrhněte aplikaci, která bude demonstrovat funkčnost těchto metod.
4. Navrženou aplikaci implementujte a proveďte experimenty s vhodnými datovými sadami.
5. Zhodnoťte dosažené výsledky a další možnosti pokračování tohoto projektu.

Literatura:

- Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Third Edition. Morgan Kaufmann Publishers, 2012, 703 p., ISBN 978-0-12-381479-1.
- Zhang, C., Zhang, S.: Association Rule Mining, Models and Algorithms. Lecture Notes in Computer Science vol. 2307, Springer Verlag, 2002, 256 p., ISBN 978-3-54-043533-4.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1-3.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Bartík Vladimír, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 11. května 2022
Datum schválení: 20. října 2021

Abstrakt

Tato bakalářská práce se zabývá získáváním víceúrovňových asociačních pravidel. Cílem této práce je zaměřit se na dostupné algoritmy pro získávání víceúrovňových asociačních pravidel a implementovat aplikaci s grafickým uživatelským rozhraním, která bude demonstrovat funkčnost těchto algoritmů. Zvoleno bylo pět algoritmů založených na algoritmu Apriori. Pomocí aplikace byly provedeny experimenty s jednotlivými algoritmy a na závěr byly výsledky experimentů porovnány a zhodnoceny.

Abstract

This bachelor thesis deals with multiple level association rules mining. The aim of this work is to focus on available algorithms for mining multiple level association rules and to implement an application with a graphical user interface that will demonstrate the functionality of these algorithms. Five algorithms based on the Apriori algorithm were chosen. Experiments with each algorithm were performed using the application and the results were compared and evaluated at the end of the thesis.

Klíčová slova

Získávání znalostí z databází, dolování dat, analýza nákupního košíku, asociační pravidla, víceúrovňová asociační pravidla, frekventované množiny, algoritmus Apriori, podpora, spolehlivost

Keywords

Knowledge discovery in databases, data mining, market basket analysis, association rules, multiple level association rules, frequent itemsets, Apriori algorithm, support, confidence

Citace

NGUYENOVÁ, Thanh Lam. *Získávání víceúrovňových asociačních pravidel.* Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vladimír Bartík, Ph.D.

Získávání víceúrovňových asociačních pravidel

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně pod vedením pana Ing. Vladimíra Bartíka, Ph.D. Uvedla jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpala.

.....
Thanh Lam Nguyenová
10. května 2022

Poděkování

Chtěla bych poděkovat svému vedoucímu práce, panovi Ing. Vladimírovi Bartíkovi, Ph.D., za odborné vedení a pomoc při vypracování této práce.

Obsah

Kapitola 1

Úvod

S rostoucím množstvím dat v úložištích po celém světě roste i potřeba tato data efektivně zpracovávat, dále analyzovat a získávat z nich nové znalosti. Protože je vzniklých dat tak velké množství, běžné metody analýzy již nejsou dostatečně výkonné a účinné. Navíc roste potřeba z těchto dat získávat skryté a na první pohled neviditelné informace, zde také nelze aplikovat běžné metody nebo nástroje. Tato potřeba tedy vedla ke vzniku nového vědního oboru, a to získávání znalostí z databází neboli dolování dat (data mining).

Získávání znalostí z databází neboli dolování dat je proces zjišťování užitečných informací ve velkých úložištích dat. Techniky dolování dat se používají k prohledávání velkých databází s cílem nalézt nové a užitečné znalosti.

Dolování dat má aplikaci v mnoha oblastech, například ve zdravotnictví, vědě, školství, bankovnictví, v oblasti trhu a marketingu, atd. Tato práce se konkrétně zaměřuje na oblast asociační analýzy. Jedním z úkolů asociační analýzy je například analýza nákupního košíku, pomocí které dokáže obchodník zjistit, jaké položky si zákazníci nejčastěji kupují společně. Tyto získané znalosti mohou obchodníkovi pomoci při například návrhu katalogů, akčních nabídek, apod. Tyto analýzou získané znalosti se vyjadřují pomocí tzv. asociačních pravidel, a tato práce se zabývá konkrétně získáváním víceúrovňových asociačních pravidel.

Cílem této práce je zaměřit se na různé dostupné metody pro získávání víceúrovňových asociačních pravidel a implementovat aplikaci s jednoduchým grafickým uživatelským rozhraním, která bude demonstrovat funkčnost těchto metod. Dále je cílem práce tyto výsledky experimentů porovnat a zhodnotit.

Práce je rozdělena do šesti kapitol. V kapitole 2 je popsaná problematika získávání znalostí z databází, proces získávání znalostí, druhy dat, které lze dolovat a základní typy dolovacích úloh. Kapitola 3 popisuje problematiku asociační analýzy a získávání asociačních pravidel, typy asociačních pravidel, proces získávání asociačních pravidel a podrobněji popisuje víceúrovňová asociační pravidla a její metody získávání. Kapitola 4 se zabývá návrhem a implementací výsledné experimentální aplikace a jejího grafického uživatelského rozhraní. V kapitole 5 jsou provedeny experimenty implementovaných algoritmů a zhodnoceny výsledky těchto experimentů. Závěrečná kapitola 6 shrnuje získané poznatky a návrhy na zlepšení práce.

Kapitola 2

Získávání znalostí z databází

Tato kapitola se zaměřuje na teoretický úvod do problematiky získávání znalostí z databází. Podkapitola 2.1 se věnuje definicí získávání znalostí z databází, motivací vzniku tohoto oboru a popisuje rozdíl mezi pojmy získávání znalostí z databází a dolování dat. V podkapitole 2.2 je popsán proces získávání znalostí. Podkapitola 2.3 se zaměřuje na druhy dat, které je možné dolovat. Důležitou částí této kapitoly je podkapitola 2.4, která popisuje základní typy dolovacích úloh. Informace v této kapitole jsou čerpány z [4], [1], [2], [7] a [8].

2.1 Získávání znalostí z databází a dolování dat

Tato podkapitola čerpá z [4], [8] a [2].

Množství dat uchovávaných v počítačových souborech a databázích v posledních letech roste velmi vysokou rychlostí. Uživatelé těchto dat od nich zároveň očekávají sofistikovanější informace. Nicméně se však ukázalo, že získávání těchto informací je extrémně náročné. Tradiční nástroje a techniky pro analýzu dat nejsou schopny odhalit požadované znalosti v obrovských objemech dat. Tento problém vedl ke vzniku tohoto oboru.

Získávání znalostí z databází (anglicky *knowlegde discovery in databases*, zkráceně *KDD*) je v [8] formálně definováno jako extrakce zajímavých, netriviálních, implicitních, dříve neznámých a potenciálně užitečných informací nebo vzorů z velkých objemů dat, kde tyto informace a vzory reprezentují znalosti získané z dat. Netriviálnost zde znamená, že tyto informace nelze získat pomocí běžných nástrojů (např. explicitním SQL dotazem nad databází), ale musí se použít nějaký sofistikovaný postup. Podobně implicitnost zde říká, že tyto informace jsou v databázi skryté a nejsou na první pohled vidět (databáze není navrhována s ohledem na to, aby tyto informace ukládala). Dále by získané znalosti měly být potenciálně užitečné, to znamená, že by měly být nějakým způsobem využitelné (např. jako podklad pro nějaké rozhodnutí nebo jako podklad pro nějakou novou hypotézu apod.)

Pojem *získávání znalostí z databází* se v praxi často zaměňuje s pojmem *dolování dat* (anglicky *data mining*). Pojem získávání znalostí z databází označuje celkový proces získávání znalostí v datech, který se skládá z několika kroků. Zatímco dolování dat je pouze jedním krokem tohoto procesu (tento proces je podrobněji popsán v sekci 2.2). Přesto se oba pojmy v současnosti nadále používají jako synonyma.

Získávání znalostí se aplikuje v mnoha oblastech. Jednou z nich je oblast analýzy trhu a marketingu. Zde může být úlohou například rozřazení zákazníků se stejnými nebo podobnými vlastnostmi do skupin (např. na základě zájmu, způsobu utrácení peněz apod.). Další typickou úlohou v této oblasti je analýza nákupního košíku. Cílem této analýzy je

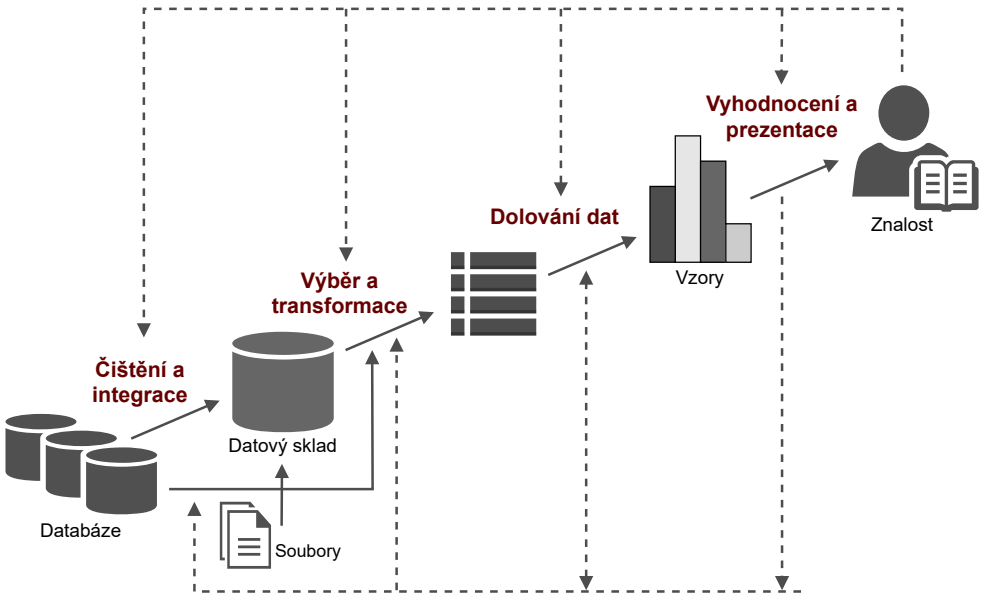
identifikovat položky v nákupním košíku, které si zákazníci často kupují společně (např. pokud si zákazník koupí počítač, kupuje si často i antivirový program). Jde tedy o hledání častých vzorů (vztahů), které jsou vyjadřovány pomocí asociačních pravidel. Tato asociační pravidla jsou předmětem této bakalářské práce a jsou podrobněji popsána v kapitole 3.2.

2.2 Proces získávání znalostí z databází

Tato podkapitola čerpá z [4] a [8].

Proces získávání znalostí z databází sestává z několika kroků, které se zpravidla v určitých iteracích opakují. Jeho grafické znázornění je možné vidět na obrázku 2.1. Jde tedy o následující kroky:

1. **Čištění dat** – cílem je odstranění šumu a nekonzistence dat.
2. **Integrace dat** – cílem je integrace dat pocházejících z několika datových zdrojů. Tento krok se často provádí společně s krokem čištění dat, protože jedním ze zdrojů nekonzistence jsou typicky data pocházející z více zdrojů. Výsledná data jsou ukládána do datových skladů.
3. **Výběr dat** – cílem je vybrat data, která jsou pro danou analytickou úlohu relevantní.
4. **Transformace dat** – cílem je transformovat data do sjednocené podoby vhodné pro dolování, např. pomocí sumarizace nebo agregace.
5. **Dolování dat** – tento krok je jádrem celého procesu získávání znalostí. Cílem je extrakce vzorů z dat určitou metodou a konkrétním algoritmem.
6. **Hodnocení modelů a vzorů** – cílem je identifikovat zajímavé vzory na základě míry užitečnosti.
7. **Prezentace znalostí** – vizualizace a prezentace nalezených znalostí.



Obrázek 2.1: Proces získávání znalostí. Převzato a upraveno z [4].

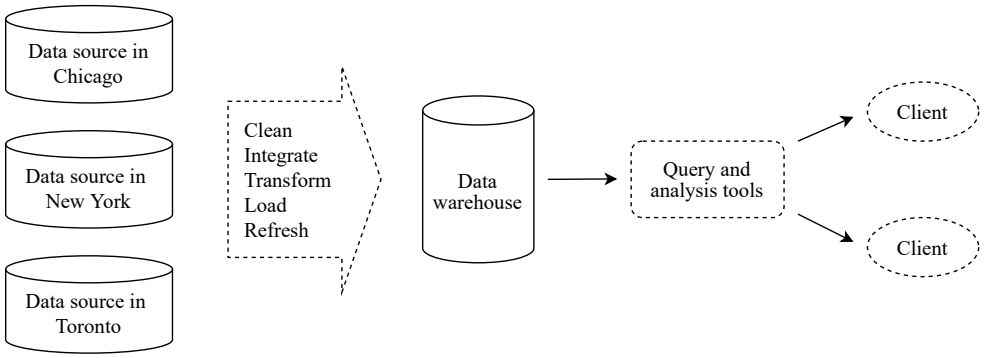
2.3 Druhy dat pro dolování

Tato podkapitola čerpá z [4] a [8].

Obecně je možné dolovat v jakémkoliv druhu dat, a to dat perzistentních uložených v nějakých úložištích nebo transientních (proudy dat). Pravděpodobně nejčastějším zdrojem dat pro dolovací úlohy jsou v současné době relační databáze. Dalšími zdroji dat mohou být datové sklady, transakční databáze a další.

1. **Relační databáze** – kolekce tabulek, které musí splňovat podmínku první normální formy (tzn. data v jednotlivých sloupcích tabulek musí být atomické). Řádky tabulky reprezentují jednotlivé záznamy a sloupce tabulky reprezentují atributy těchto záznamů. Každý záznam v tabulce je jednoznačně identifikován primárním klíčem. K relačním datům lze přistupovat pomocí databázových dotazů napsaných v relačním dotazovacím jazyce (např. SQL). Daný dotaz je transformován do množiny relačních operací, jako je spojení, výběr a projekce, a následně je optimalizován pro efektivní zpracování.

2. **Datové sklady** – komplexní úložiště, ve kterém jsou uložená data z více zdrojů. Tato data jsou zde uložena z historické perspektivy (např. data za posledních 5 let) a jsou typicky sumarizovaná. Datový sklad je obvykle modelován jako tzv. multidimenzionální datová kostka. Datová kostka je multidimenzionální datová struktura, u které každá dimenze odpovídá atributu nebo skupině atributů ve schématu databáze a každá buňka obsahuje agregované údaje (např. počty prodaných kusů).



Obrázek 2.2: Koncepce datového skladu. Převzato z [4].

3. **Transakční databáze** – soubor, ve kterém každý záznam reprezentuje jednu transakci. Nejde o transakce z databází, ale o obchodní transakce (např. nákup zboží zákazníkem). Transakce je obvykle tvořená jednoznačným identifikátorem a seznam položek, které transakci tvoří. Transakční databáze může obsahovat i další tabulky například s informacemi o prodávaném zboží nebo zákaznících apod.

trans_ID	list_of_items_IDs
T100	I1, I3, I8, I16
T200	I2, I8
...	...

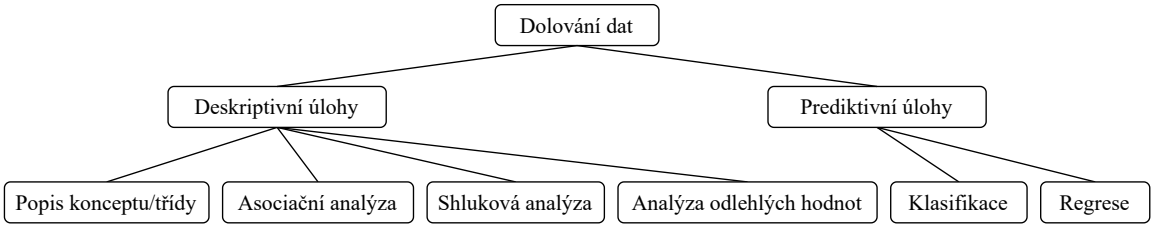
Dalšími typy zdrojů dat mohou být například objektově-relační databáze, databáze časových řad, multimediální databáze, textové databáze, proudy dat, web, atd.

2.4 Typy dolovacích úloh

Tato podkapitola čerpá z [1], [4] a [8].

Dolovací úlohy je možné rozdělit do dvou hlavních kategorií:

1. **Deskriptivní** – úlohy, které charakterizují obecné vlastnosti analyzovaných dat v databázi. Mezi deskriptivní úlohy patří například shluková analýza nebo asociační analýza.
2. **Prediktivní** – úlohy, které využívají analýzy současných dat k předpovědi budoucích chování. Mezi prediktivní úlohy patří například klasifikace a regrese.



Obrázek 2.3: Příklady dolovacích úloh.

2.4.1 Popis konceptu/třídy

Cílem této deskriptivní úlohy je asociovat data s určitou třídou nebo konceptem. Například firma prodávající elektroniku může mít třídy položek *počítač* a *tiskárna* a koncepty zákazníků *utráceč* a *rozpočtář*. Je tedy užitečné charakterizovat jednotlivé třídy a koncepty nějakým souhrnným, stručným a přesto dostatečně přesným způsobem. Takový popis je možné získat jedním ze dvou následujících způsobů:

1. **Charakterizací dat** – cílem je sumarizovat obecné vlastnosti analyzované třídy. Data analyzované třídy lze zpravidla získat jednoduchými dotazy nad databází. Výstupem charakterizace dat mohou být prezentovány v různých formách, např. koláčové grafy, sloupcové grafy, křivky, apod.
2. **Diskriminací dat** – cílem je porovnat obecné vlastnosti analyzované třídy s obecnými vlastnostmi jiné referenční třídy nebo množiny třídy. Hledají se tedy atributy, ve kterých se tyto třídy liší. Výstupy diskriminace jsou podobné jako u charakterizace dat, ale musí navíc obsahovat porovnání těchto tříd.

2.4.2 Frekventované vzory, asociace a korelace

Frekventované vzory, jak vyplývá z názvu, jsou vzory, které se vyskytují v datech často. Existuje mnoho různých druhů frekventovaných vzorů, jako např. frekventované množiny, frekventované podsekvence (též známé jako sekvenční vzory) a frekventované podstruktury (podgrafy, podstromy, atd.). Získávání frekventovaných množin vede k odhalení zajímavých asociací a korelací v datech.

Odhalením zajímavých asociací v datech se zabývá asociační analýza, u které jsou výsledkem asociační pravidla. Příklad asociačního pravidla vypadá následovně:

$$kupuje(X, \text{“počítač”}) \Rightarrow kupuje(X, \text{“software”})[podpora = 1\%, spolehlivost = 50\%],$$

kde proměnná X označuje zákazníka. Toto asociační pravidlo říká, že zákazníci, kteří si koupí počítač, si často koupí i software. *Podpora* a *spolehlivost* jsou dvě nejčastěji používané míry, které vyjadřují významnost zastoupení příslušného frekventovaného vzoru v analyzovaných datech a jsou to míry, které se používají k poměřování zajímavosti dolovaných asociačních pravidel. Tato asociační pravidla jsou předmětem této bakalářské práce a podrobněji se jimi věnuje kapitola 3.1 níže.

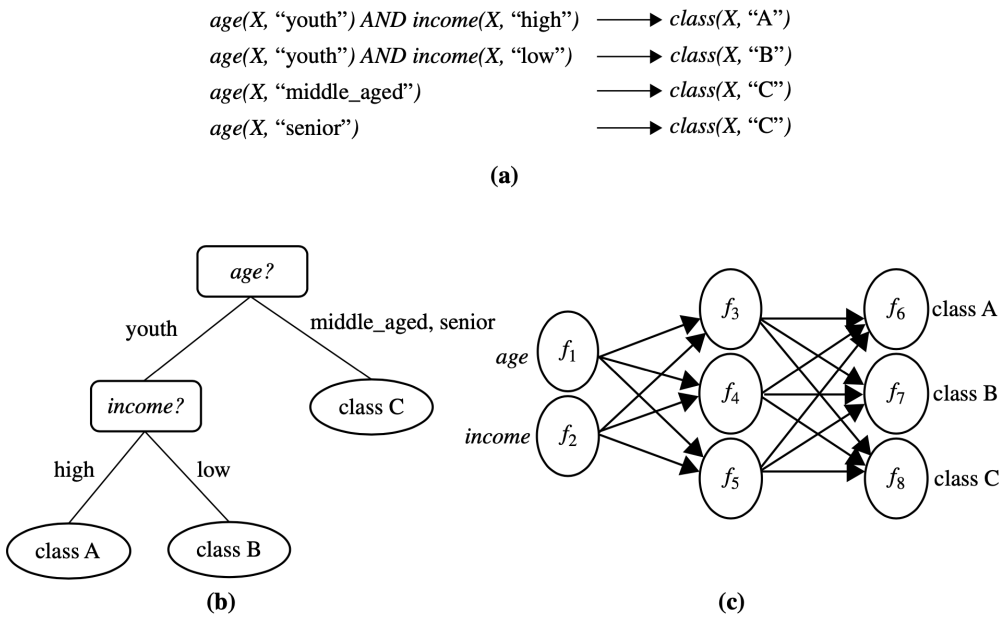
2.4.3 Klasifikace a regrese

Klasifikace je proces hledání modelu, který popisuje a současně rozlišuje třídy dat. Tento model lze získat analýzou trénovacích sad (tj. data objektů, u kterých je třída zařazení známá) a poté je použit pro predikci třídy objektu, u kterého zařazení není známé.

Proces klasifikace se skládá ze tří kroků:

1. **Trénování (učení)** – za použití trénovacích dat je vytvořen klasifikační model
2. **Testování** – hodnocení kvality vytvořeného modelu použitím testovacích dat
3. **Aplikace** – použití modelu pro klasifikaci objektu, jehož třída je neznámá

Klasifikační model může mít různou podobu, například jako klasifikační pravidla (if-then), rozhodovací stromy nebo neuronové sítě (viz obrázek 2.4). Existuje ale řada dalších metod konstrukce klasifikačních modelů.

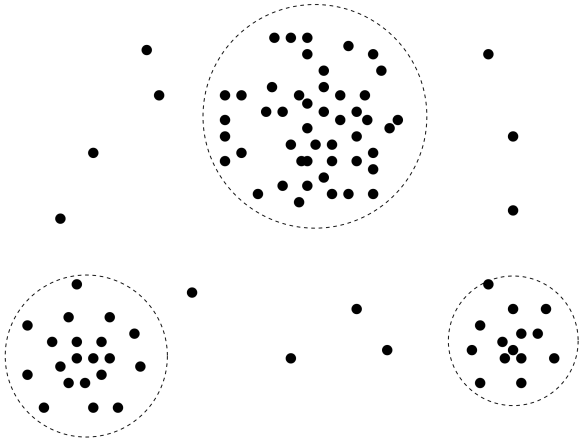


Obrázek 2.4: Formy reprezentace klasifikačního modelu: a) klasifikační pravidla, b) rozhodovací strom, c) neuronová síť. Převzato z [4].

Klasifikace se používá k predikci hodnot kategorických (diskrétních, neuspořádaných) hodnot tříd. Pro hodnoty spojitých atributů se používá regrese. To znamená, že v tomto případě se predikuje chybějící nebo nedostupnou numerickou hodnotu. Nejčastější metodou numerické predikce je *regresní analýza*.

2.4.4 Shluková analýza

Narozdíl od klasifikace a regrese, které při vytváření klasifikačního modelu analytují datové objekty, jejichž přiřazení do tříd je známé, shluková analýza analyzuje datové objekty bez znalosti jejich přiřazení do tříd. Cílem shlukování je nalézt třídy objektů, které mají co nejvíce společného a naopak třídy objektů, které se od ostatních tříd co nejvíce liší. Cílem je tedy maximilizovat podobnost objektů v rámci shluku, ale minimalizovat podobnost objektů s objekty mimo tento shluk. Takto vytvořené třídy mají podobu homogenních skupin, tzv. shluků (viz obrázek 2.5).



Obrázek 2.5: Příklad 2D grafu výsledku shlukové analýzy. Převzato z [4].

2.4.5 Analýza odlehlých hodnot

Někdy mohou zajímavé vzory představovat ne takové hodnoty, které se v analyzovaných datech vyskytují často, ale takové, které se významně odlišují od ostatních. Takové datové objekty se nazávají odlehlé objekty a jejich hledáním se zabývá analýza odlehlých hodnot. V praxi se analýza odlehlých hodnot může využít při detekci podvodů, například při odhalení podvodného použití kreditních karet, a to detekcí neobvyklých vysokých plateb nebo neobyklých lokací těchto plateb apod.

Kapitola 3

Získávání asociačních pravidel

Tato kapitola se zaměřuje na problematiku získávání asociačních pravidel. Podkapitola 3.1 se zabývá úvodem do problematiky asociační analýzy. Podkapitola 3.2 definuje asociační pravidla a popisuje jejich hlavní parametry - podporu a spolehlivost. V podkapitole 3.3 jsou uvedeny a popsány základní typy asociačních pravidel a jejich rozdělení. Proces získávání asociačních pravidel je rozdělen do dvou fází, první fází se zabývá podkapitola 3.4 a druhou fází podkapitola 3.5. Kapitola 3.6 se zaměřuje na víceúrovňová asociační pravidla, která jsou hlavním předmětem této práce. Na závěr jsou v podkapitole 3.6.1 popsány různé metody pro získávání víceúrovňových asociačních pravidel. Informace v této kapitole jsou čerpány z [4], [8] a [7].

3.1 Úvod do problematiky

Tato podkapitola čerpá z [4] a [7].

Asociační analýza je jedním z nejvýznamnějších metod v dolování dat. Cílem asociační analýzy je objevit zajímavé asociace nebo korelace nad velkými množinami datových položek. Řada obchodních společností shromažďuje obrovské množství dat ze svých každodenních operací. Například v supermarketech se denně shromažďuje obrovské množství transakčních záznamů zákazníků. Nalezení zajímavých asociací nad těmito transakčními záznamy může pomoci v procesu obchodního rozhodování, jako je návrh katalogů, akčních nabídek, marketingové propagace nebo rozmístění zboží v obchodě.

Typickým příkladem pro získávání asociačních pravidel je *analýza nákupního košíku*. Tento proces analyzuje a hledá asociace mezi různými položkami zboží, které zákazníci vkládají do svého nákupního košíku. Jinými slovy tedy zjišťuje, jaké zboží si zákazníci často kupují společně.

Kromě analýzy nákupního košíku se získávání asociačních pravidel v současné době aplikace v mnoha dalších oblastech, například v bioinformatice, medicíně (lékařské diagnózy), Web miningu, analýze vědeckých dat, atd.

3.2 Asociační pravidla

Tato podkapitola čerpá z [4] a [8].

Jak bylo zmíněno výše, cílem asociační analýzy je najít zajímavé asociace (vztahy) nad velkými množinami datových položek. Tyto asociace jsou následně reprezentovány pomocí tzv. *asociačních pravidel* nebo frekventovanými množinami. Příklad asociačního pravidla může vypadat následovně:

$$kupuje(X, \text{“klávesnice”}) \Rightarrow kupuje(X, \text{“myš”})[podpora = 2\%, spolehlivost = 60\%].$$

Toto pravidlo říká, že mezi položkami *klávesnice* a *myš* existuje silný vztah, protože zákazníci, kkteří si koupí klávesnici, si často koupí i myš. Proměnná X zde označuje zákazníka. Podpora 2% zde znamená, že ve 2% transakcí byly tyto dvě položky společně. Spolehlivost 60% znamená, že 60% zákazníků, kteří si koupili klávesnici, si také koupili myš. Asociační pravidlo se považuje za zajímavé, je-li splněna podmínka minimální podpory a minimální spolehlivosti.

Základní definice asociačního pravidla Nechť $I = (i_1, i_2, i_3, \dots)$ je množina položek. Dále množina transakcí D je množina transakcí, kde každá transakce T je množina položek taková, že $T \subseteq I$. Ke každé transakci přísluší unikátní identifikátor TID. Nechť A je množina položek. Říkáme, že transakce T obsahuje A, pokud $X \subseteq T$. Asociační pravidlo je implikace tvaru $A \Rightarrow B$, kde $A \subset T$, $B \subset T$ a $A \cap B = \varnothing$.

Podpora a spolehlivost *Podpora* (anglicky *support*) a *spolehlivost* (anglicky *confidence*) jsou základními charakteristikami asociačních pravidel.

Pravidlo $A \Rightarrow B$ má podporu s v množině transakcí D, jestliže $s\%$ transakcí v D obsahuje množinu položek $A \cup B$.

Pravidlo $A \Rightarrow B$ má spolehlivost c, jestliže $c\%$ transakcí, které obsahují A obsahuje také B.

S využitím pravděpodobnosti lze tedy napsat:

$$podpora\,(A \Rightarrow B) = P\,(A \cup B) \tag{3.1}$$

$$spolehlivost\,(A \Rightarrow B) = P\,(A|B) = \frac{podpora\,(A \cup B)}{podpora\,(A)} \tag{3.2}$$

Asociační pravidla, která splňují podmínku minimální podpory a minimální spolehlivosti se označují jako *silná asociační pravidla.*

Množina položek je množina obsahující jednu nebo více položek. Množina položek, která obsahuje k položek se označuje jako k-množina. Například množina $\{klávesnice, myš\}$ je 2-množina. Množina položek, která má podporu vyšší než uživatelem zadaná minimální hodnota podpory, se označuje jako *frekventovaná množina.*

Proces získávání asociačních pravidel Celý proces získávání asociačních pravidel se dělí do dvou následujících kroků:

1. **Nalezení frekventovaných množin** – nalezení takových množin, které splňují podmínku minimální podpory.
2. **Generování silných asociačních pravidel z frekventovaných množin** – tato pravidla musí splňovat podmínku minimální podpory a minimální spolehlivosti.

3.3 Typy asociačních pravidel

Tato podkapitola čerpá z [4] a [8].

Asociační pravidla mohou být rozdělena podle následujících kritérií:

- **Podle typu hodnot v pravidlech**
  - **Booleovská asociační pravidla** – pravidlo obsahuje asociace mezi přítomností nebo nepřítomností položek.

$$koupí(X, \text{“počítač”}) \Rightarrow koupí(X, \text{“tiskárna”}) \tag{3.3}$$

  - **Kvantitativní asociační pravidla** – pravidlo popisuje asociace mezi kvantitativními položkami nebo atributy. V těchto pravidlech jsou kvantitativní hodnoty položek nebo atributů obvykle rozděleny do intervalů.

$$věk(X, \text{“}20 \ldots 29\text{”}) \wedge příjem(X, \text{“}40000 \ldots 45000\text{”}) \Rightarrow koupí(X, \text{“}iPhone\text{”}) \tag{3.4}$$

- **Podle dimenzí obsažených v pravidlech**
  - **Jednodimezionální asociační pravidla** – pravidlo obsahující pouze jeden atribut. Příkladem může být pravidlo 3.3 zmíněné výše, u kterého je jediným atributem atribut *koupí*.
  - **Multidimenzionální asociační pravidla** – pravidlo obsahující dvě a více atributů. Příkladem tohoto asociačního pravidla může být pravidlo 3.4, které obsahuje tři atributy *věk*, *příjem* a *koupí*.
- **Podle úrovní abstrakce v pravidlech**
  - **Jednoúrovňová asociační pravidla** – pravidlo pracující s jedinou úrovní abstrakce, příkladem je pravidlo 3.3.
  - **Víceúrovňová asociační pravidla** – pravidlo pracující s položkami na různých úrovních abstrakce. Tyto jednotlivé položky tvoří hierarchii, tzv. konceptuální hierarchii, která je podrobněji popsána v sekci 3.6 níže.

$$koupí(X, \text{“počítač”}) \Rightarrow koupí(X, \text{“barevná laserová tiskárna”}) \tag{3.5}$$

3.4 Získávání frekventovaných množin

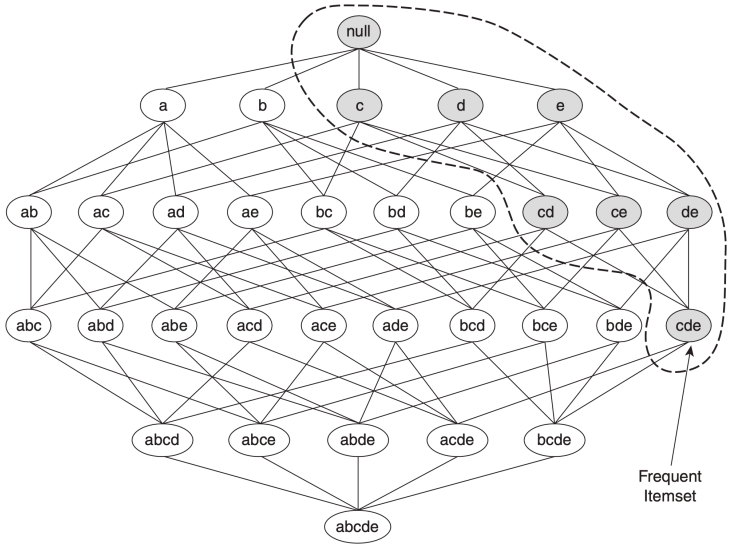
Tato podkapitola čerpá z [4], [8] a [7].

Jak bylo zmíněno výše v 3.2, prvním krokem procesu získávání asociačních pravidel je získávání frekventovaných množin. Nejznámějším a nejvíce používaným algoritmem pro získávní frekventovaných množin je algoritmus Apriori, který navrhli R. Agrawal a R. Srikant.

3.4.1 Algoritmus Apriori

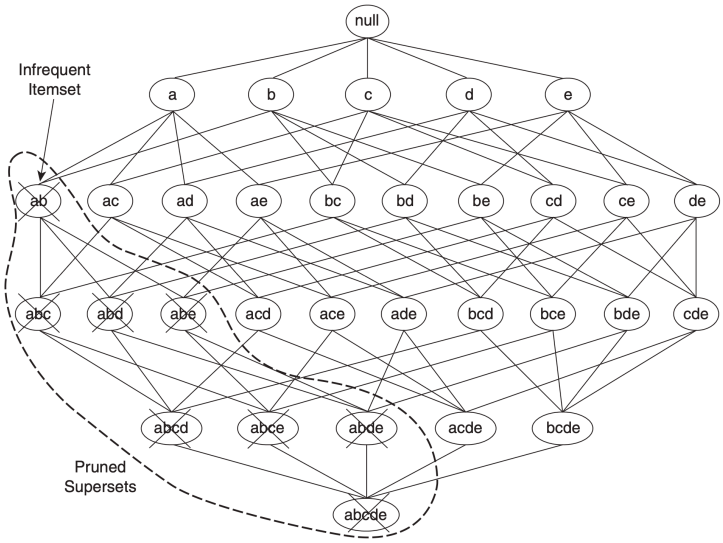
Název algoritmu vychází z faktu, že využívá předchozí znalost o dříve získaných frekventovaných množinách. V každé iteraci jsou získané frekventované k-množiny použity pro generování $(k+1)$-množin. Nejdříve dochází k vytvoření 1-množin průchodem databází, při kterém se vybírají položky, které splňují podmínku minimální podpory. Výsledkem je tedy množina, která se označuje jako L_1. Dále je L_1 použita k vytvoření L_2, která je dále použita k vytvoření L_3, atd. Tento postup se opakuje do té doby, kdy se nepodaří najít žádná další frekventovaná množina. Každá iterace tedy vyžaduje jeden průchod databází.

Apriori vlastnost K vyšší efektivitě algoritmu Apriori se využívá tzv. *Apriori vlastnost*. Tato vlastnost říká, že každá podmožina frekventované množiny musí být také frekventovaná. Například, je-li množina $\{c, d, e\}$ frekventovanou množinou, potom transakce, které obsahují množinu $\{c, d, e\}$ musí také obsahovat její podmnožiny $\{c, d\}$, $\{c, e\}$, $\{d, e\}$, $\{c\}$, $\{d\}$ a $\{e\}$. Z toho tedy vyplývá, je-li množina $\{c, d, e\}$ frekventovaná, musí být i všechny její podmnožiny frekventované. Tento příklad je graficky znázorněn na obrázku 3.1.



Obrázek 3.1: Ilustrace Apriori vlastnosti. Převzato a upraveno z [7].

A naopak, není-li množina $\{a, b\}$ frekventovanou množinou, potom nejsou frekventované ani všechny její nadmnožiny a mohou být tedy ignorovány. Tento příklad je graficky znázorněn na obrázku 3.2.



Obrázek 3.2: Ilustrace Apriori vlastnosti. Převzato a upraveno z [7].

Apriori algoritmus Algoritmus Apriori se skládá ze dvou kroků:

1. **Spojovací krok** – K nalezení L_k (všech frekventovaných množin), kandidáti na frekventované množiny C_k jsou vygenerováni spojením množin z L_{k-1}. Nechť l_1 a l_2 jsou množiny L_{k-1}. Pro efektivní implementaci se předpokládá, že položky v množině jsou lexickograficky uspořádané. Ke spojení dvou $(k-1)$-množin dojde, pokud jejich prvních $k-2$ prvků je shodných. Výsledná množina vzniklá spojením l_1 a l_2 bude mít prvky $l_1[1], l_1[2], \ldots, l_1[k-1], l_2[k-1]$. $l_1[i]$ je i-tý prvek množiny l_1 a musí platit, že $l_1[k-1] < l_2[k-1]$.

2. **Vylučovací krok** – množina kandidátů C_k je nadmnožinou L_k, to znamená, že její prvky mohou, ale nemusí být frekventované. Zjišťování výskytu každého kandidáta v databázi vyžaduje průchod databází, to je velmi neefektivní, protože databáze může být příliš velká a může proto vyžadovat příliš náročné výpočty. Pro zredukování databáze se využívá Apriori vlastnost. Žádná $(k-1)$-množina, která není frekventovaná, nemůže být podmnožinou frekventované k-množiny. Tedy, pokud některá $(k-1)$-podmnožina kandidátní k-množiny není v L_{k-1}, pak není frekventovaná ani kandidátní k-množina a lze ji odstranit z C_k. Toto vyhledávání může být zrychleno pomocí hašovacího stromu pro frekventované množiny.

Zde je ukázka pseudokódu pro algoritmus Apriori:

Data: databáze D, hodnota minimální podpory *min_supp*
Result: L – frekventované množiny v databázi D

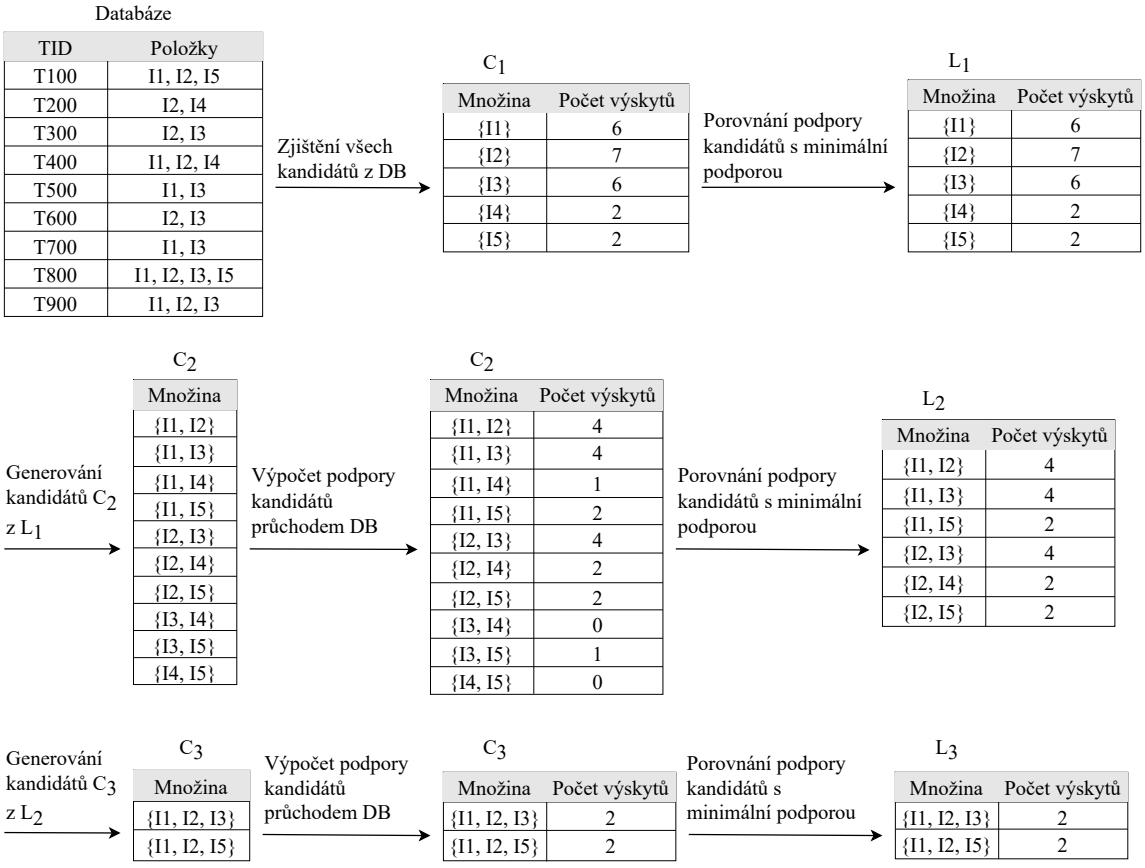
```
1  L_1 = nalezni_frekventované_1-množiny(D);
2  for (k=2; L_{k-1} ≠ ∅; k++) do
3  |   C_k = apriori_gen(L_{k-1});
4  |   foreach transakce t ∈ D do
5  |   |   C_t = subset(C_k, t);
6  |   |   foreach kandidát c ∈ C_t do
7  |   |   |   c.počet_výskytů++;
8  |   |   end
9  |   end
10 |   L_k = {c ∈ C_k | c.počet_výskytů ≥ min_supp};
11 end
12 return L = ∪_k L_k;
```

V prvním kroku jsou vygenerovány frekventované 1-množiny. Dále se v jednotlivých iteracích spouští funkce *Apriori_gen*, která provádí spojovací a vylučovací fázi, tedy generuje kandidáty. Poté se prochází databáze a pro každého kandidáta se spočítá počet jeho výskytů. Kandidáti, jejichž podpora je vyšší než minimální, jsou uloženi do L_k.

Procedura *Apriori_gen* vypadá následovně:

```
1  foreach množina l_1 ∈ L_{k-1} do
2      foreach množina l_2 ∈ L_{k-1} do
3          if
            (l_1[1] = l_2[1]) ∧ (l_1[2] = l_2[2]) ∧ ··· ∧ (l_1[k-2] = l_2[k-2]) ∧ (l_1[k-1] = l_2[k-1])
           then
4              c = l_1 ⋈ l_2;
5              if má_nefrekventovanou_množinu(c, L_{k-1}) then
6                  odstraň c;
7              else
8                  přidej c do C_k;
9              end
10         end
11     end
12 end
13 return C_k;
```

Postup generování frekventovaných množin pomocí algoritmu Apriori je ukázán na jednoduchém příkladu, který je graficky znázorněn níže. Předpokládá se, že minimální podpora je 22% (tzn. počet výskytů je roven 2).

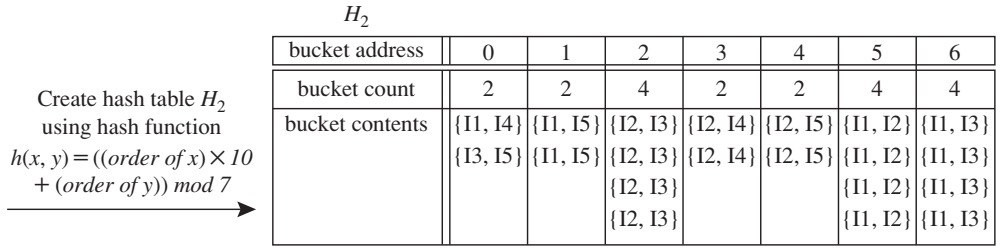


Obrázek 3.3: Příklad algoritmu Apriori. Převzato a upraveno z [4].

3.4.2 Zvýšení efektivity algoritmu Apriori

Tato sekce čerpá z [4].

- **Hašovací přístup** – jde o hašování položek do příslušných "košů". Cílem je tohoto přístupu je redukce kandidátních k-množin C_k, kde $k > 1$. Při získávání frekventovaných 1-množin průchodem transakcí databáze se vygenerují všechny 2-množiny pro každou transakci a tyto množiny se přiřadí do košů hašovací tabulky. Pokud je počet položek u některého koše menší než minimální podpora, pak 2-množiny v tomto koši jsou nefrekventované a dále se s nimi nepočítá. Tento hašovací přístup dokáže podstatně redukovat kandidátní množiny, zvláště když $k = 2$.



Obrázek 3.4: Příklad hašovacího přístupu – hašovací tabulka. Převzato z [4].

- **Redukce transakcí** – transakce, která neobsahuje žádnou frekventovanou k-množinu nemůže obsahovat frekventované $(k + 1)$-množiny. Proto se taková transakce může odstranit nebo ignorovat a dále se s transakcí nebude pracovat.

- **Rozdělení dat** – tento přístup vyžaduje pouze dva průchody databází a skládá se ze dvou fází. V první fázi se databáze rozdělí do n částí. V každé části jsou zjištěny tzv. lokální frekventované množiny, které se poté stanou kandidáty pro získávání tzv. globálních frekventovaných množin, které jsou nakonec výsledkem.

- **Vzorkování** – zde jde o použití náhodně vybraného vzorku z dat a získání frekventovaných množin z něj. Výhodou tohoto přístupu je, že je velmi efektivní, ale naopak nevýhodou je, že není velmi přesný. Aby byla přesnost vyšší, musí se počítat s podporou nižší než je daná minimální podpora. Tím se ale získají i množiny, které jsou frekventované. Proto se musí posléze ověřit podporu získaných frekventovaných množin na celé databázi.

- **Dynamické počítání množin** – zde se kandidátní množiny získávají v průběhu čtení databáze. Přidává kandidáta pouze tehdy, když všechny jeho podmnožiny vypadají, že budou frekventované.

3.5 Generování asociačních pravidel

Tato podkapitola čerpá z [4] a [8].

Generování asociačních pravidel je druhým krokem celého procesu získávání asociačních pravidel. Toto generování se provádí pomocí rovnice pro výpočet spolehlivostí:

$$spolehlivost\,(A \Rightarrow B) = P\,(A|B) = \frac{podpora\,(A \cup B)}{podpora\,(A)}$$

- Pro každou frekventovanou množinu l se vygenerují všechny její neprázdné podmnožiny.
- Pro každou podmnožinu s se vygeneruje pravidlo $s \Rightarrow (l - s)$ a podle rovnice se vypočítá jeho spolehlivost. Pokud je jeho spolehlivost vyšší než minimální, pak je pravidlo silné.

Protože jsou pravidla generována z frekventovaných množin, automaticky splňují podmínku minimální podpory. Na následujícím příkladu bude ukázáno generování asociačních pravidel, a to konkrétně pro frekventovanou množinu $\{I1, I2, I5\}$ z předchozího příkladu (viz obrázek 3.3).

Neprázdnými podmnožinami jsou $\{I1, I2\}, \{I1, I5\}, \{I2, I5\}, \{I1\}, \{I2\}, \{I5\}$. Budou vygenerována tato asociační pravidla:

$$I1 \wedge I2 \Rightarrow I5; spolehlivost = 2/4 = 50\%$$
$$I1 \wedge I5 \Rightarrow I2; spolehlivost = 2/2 = 100\%$$
$$I2 \wedge I5 \Rightarrow I1; spolehlivost = 2/2 = 100\%$$
$$I1 \Rightarrow I2 \wedge I5; spolehlivost = 2/6 = 33\%$$
$$I2 \Rightarrow I1 \wedge I5; spolehlivost = 2/7 = 29\%$$
$$I5 \Rightarrow I1 \wedge I2; spolehlivost = 2/2 = 100\%$$

Předpokládá se, že minimální spolehlivost je rovna 70%. Proto jsou výstupem tři silná asociační pravidla, a to druhé, třetí a poslední.

3.6 Víceúrovňová asociační pravidla

Tato podkapitola čerpá z [4], [3] a [8].

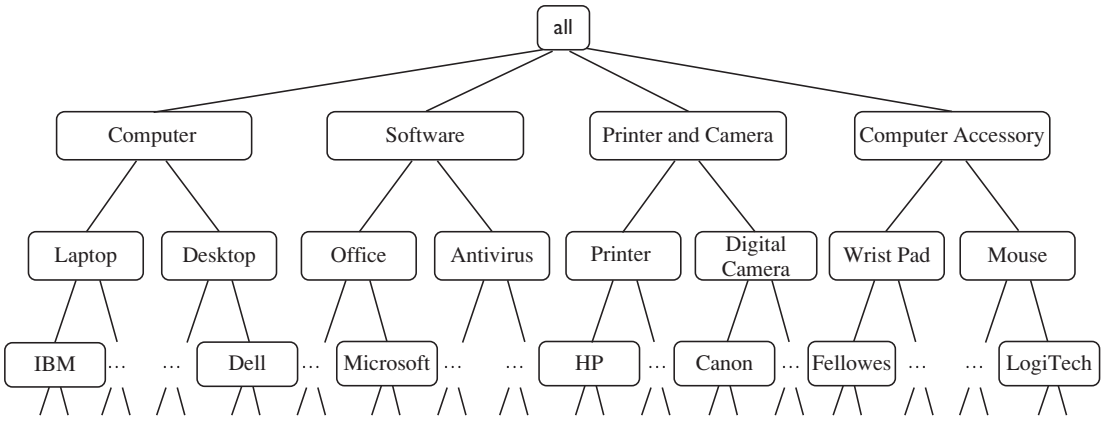
Předchozí studia se zabývala převážně získáváním jednoúrovňových asociačních pravidel. Nicméně, získáváním asociačních pravidel na více úrovních může vést k nalezení konkrétnějších a zajímavějších znalostí z dat. Kvůli řídkosti dat na nejnižší úrovni abstrakce je složitější nad položkami na této úrovni najít zajímavé asociaciace.

Například v transakční databázi 3.1 je transakcí, ve kterých se společně vyskytují např. položky *Myš Logitech* a *Počítač IBM*, minimální počet, proto je hledání asociačních pravidel na nižší úrovni obtížnější. Naopak na vyšší úrovni abstrakce u položek *Příslušenství* a *Počítač* je pravděpodobnost získání zajímavějších asociačních pravidel vyšší.

TID	Položky
T100	Počítač IBM, ČB tiskárna SONY
T200	Výukový SW Microsoft, Klávesnice Ergoway
T300	Myš Logitech, Klávesnice Ergoway
T400	Počítač IBM, Ekonomický SW Microsoft
T500	Počítač IBM

Tabulka 3.1: Příklad transakční databáze

K reprezentaci položek na jednotlivých úrovních abstrakce slouží tzv. *konceptuální hierarchie*. Jedná se o strom, který mapuje položky na nižší úrovni na koncepty vyšší úrovně. Příklad konceptuální hierarchie je možné vidět na obrázku 3.5 níže.



Obrázek 3.5: Příklad konceptuální hierarchie. Převzato z [4].

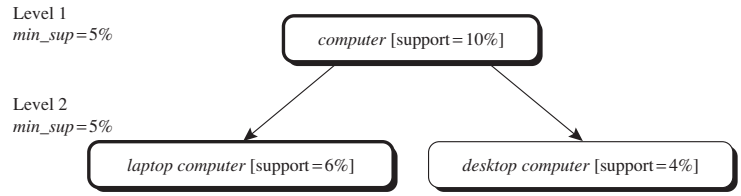
3.6.1 Metody získávání víceúrovňových asociačních pravidel

Tato sekce čerpá z [8].

Základní metodou získávání víceúrovňových asociačních pravidel je procházení hierarchií shora dolů. Tato metoda generuje frekventované množiny na každé úrovni abstrakce stejným způsobem jako při generování frekventovaných množin pro jednoúrovňová asociační pravidla dokud nenalezne žádné další frekventované množiny. Pro generování těchto frekventovaných množin se může použít algoritmus Apriori.

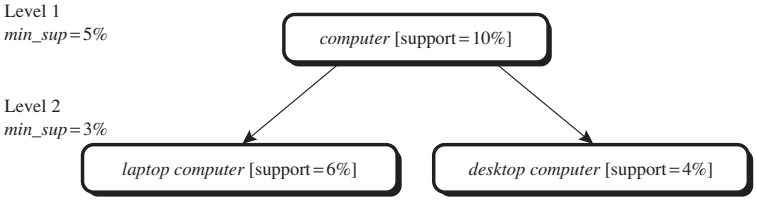
Existují dva přístupy, a to s použitím konstantní podpory nebo redukované podpory.

1. **Použití konstantní podpory** – jedná se o použití stejné hodnoty minimální podpory pro každou úroveň abstrakce konceptuální hierarchie. Výhodou je, že se jedná o velmi jednoduchou metodu. Naopak nevýhodou je, že položky na nejnižší úrovni konceptuální hierarchie se nevyskytují tak často jako položky na vyšších úrovních abstrakce. Mohou být proto při zvolení příliš vysoké hodnoty minimální podpory ztraceny zajímavé asociace na nejnižší úrovni. Naopak při zvolení příliš nízké hodnoty minimální podpory může dojít na vyšších úrovních k získání mnoho nezajímavých asociačních pravidel. Pro zjednodušení lze využít vlastnosti, že žádný následník nemůže být frekventovanou položkou, pokud jeho předchůdce není frekventovanou položkou.



Obrázek 3.6: Získávání víceúrovňových pravidel s konstantní podporou. Převzato z [4].

2. **Použití redukované podpory** – jedná se o použití redukované hodnoty minimální podpory pro nižší úrovně abstrakce konceptuální hierarchie. Pro tento přístup existuje několik variací:



Obrázek 3.7: Získávání víceúrovňových pravidel s redukovanou podporou. Převzato z [4].

- **Nezávisle na úrovních** – jedná se o procházení celé konceptuální hierarchie, bez ohledu na již získané frekventované množiny. To znamená, že každý uzel je testován, aniž by se zjišťovalo, zda je jeho otcovský uzel frekventovaný. Tato variace je nejjednodušší ze čtyř zmíněných, ale její nevýhodou je, že vede k testování velkého množství nefrekventovaných položek.
- **Filtrování úrovní k-množinami** – zde je množina na i-té úrovni testována pouze v případě, pokud je odpovídající množina na vyšší úrovni $i-1$ frekventovaná. Pokud je například z obrázku 3.5 množina $\{Computer, Software\}$ frekventovaná, pak i množiny $\{Laptop, Desktop\}$ a $\{Office, Antivirus\}$ budou testovány, v opačném případě nebudou brány v ohled. Nevýhodou tohoto přístupu je, že je zde omezení příliš silné, proto dochází ke ztrátě zajímavých asociací.
- **Filtrování úrovní jednou položkou** – zde je položka na i-té úrovni testována pouze v případě, pokud je otcovská položka na vyšší úrovni $i-1$ frekventovaná. Jinými slovy, pokud je položka frekventovaná, pak jsou testováni jeho následníci, v opačném případě nejsou brány v ohled. Tento přístup je kompromisem mezi přístupem *Nezávisle na úrovních* a *Filtrování úrovní k-množinou*. Nevýhodou u tohoto přístupu je, že některé položky, které by byly frekventované díky snížení podpory, jsou filtrováním ignorovány a dochází tedy ke ztrátě asociací.
- **Kontrolované filtrování úrovní jednou položkou** – u této variace se kromě hodnoty minimální podpory navíc zavádí tzv. práh pro zanoření. Hodnota tohoto prahu je zpravidla mezi minimálními podporami pro aktuální i-tou úroveň a pro nižší $i-1$ úroveň, do které se bude zanořovat. Je-li podpora položky vyšší než hodnota prahu, pak je testován i jeho následník i přesto, že položka není frekventovaná. Tento přístup oproti *Filtrování úrovní jednou položkou* umožňuje vznik asociací i z položek, které nemají frekventované předchůdce.

Kapitola 4

Návrh a implementace

Tato kapitola popisuje návrh a implementaci aplikace pro získávání víceúrovňových asociačních pravidel. Podkapitola 4.1 popisuje návrh aplikace a požadavky na ní. V této podkapitole jsou také popsány vybrané knihovny, použité technologie a návrh grafického uživatelského rozhraní. Podkapitola 4.2 popisuje implementaci aplikace a jejího grafického uživatelského rozhraní.

4.1 Návrh aplikace

Cílem této práce je implementovat aplikaci s jednoduchým grafických uživatelským rozhraním, které bude pomocí pěti algoritmů popsané v teoretické části práce v 3.6.1 získávat víceúrovňová asociační pravidla z transakční databáze. Aplikace by měla splňovat následující požadavky:

- jednoduché uživatelské rozhraní
- dolovat na jedné až tří úrovních konceptuální hiearchie
- výběr výchozí nebo vlastní databáze
- zobrazení výsledných asociačních pravidel
- zobrazení výsledných frekventovaných množin
- export výsledných asociačních pravidel

Program bude napsaný v programovacím jazyce Python [5] a pro implementaci procesu získávání asociačních pravidel jsou využity knihovny, které jsou popsány níže.

4.1.1 Pandas

`Pandas` je jedna z nejpopulárnějších a nejpoužívanějších knihoven při analýze dat a úlohách strojového učení. Je postavena na knihovně `Numpy`, která poskytuje podporu pro práci s vícerozměrnými poli. Základní struktury, které knihovna `Pandas` nabízí, jsou například `Series` a `DataFrame`. Struktura typu `Series` je homogenní a jednorozměrná. Struktura typu `DataFrame` je heterogenní a dvourozměrná a zobrazuje se jako tabulka. Tato knihovna navíc obsahuje velké množství funkcí, které umožňují efektivní práci s těmito strukturami.

4.1.2 Mlxtend

Tato sekce čerpá z [6].

Pro získávání víceúrovňových asociačních pravidel byla vybrána knihovna `mlxtend`, autorem této knihovny je Sebastian Raschka. `Mlxtend` je knihovna, která implementuje různé základní algoritmy a nástroje pro práci v oblasti strojového učení a dolování dat. Knihovna je implementovaná v jazyce Python. Kromě algoritmů pro získávání frekventovaných vzorů a asociačních pravidel obsahuje knihovna algoritmy pro například sekvenční výběr vlastností nebo algoritmy pro klasifikaci a regresi.

Jak bylo zmíněno výše v podkapitole 3.2, proces získávání asociačních pravidel se skládá ze dvou fází, a to získávání frekventovaných množin a generování asociačních pravidel z těchto získaných množin. Pro získávání frekventovaných množin je použit algoritmus Apriori, který je podrobně popsán výše v podkapitole 3.4.

První fázi procesu implementuje metoda `apriori()`, která vyžaduje na vstupu tzv. one-hot kódovanou tabulku `DataFrame` (knihovna *pandas*) . Sloupce v této tabulce reprezentují jednotlivé položky a řádky reprezentují jednotlivé transakce. Přítomnost položky v transakci se značí hodnotou `1` nebo booleovskou hodnotou `True` a naopak nepřítomnost položky v transakci se značí hodnotou `0` nebo `False`. Příklad takové tabulky je možné vidět níže v tabulce 4.1. Dále metoda na vstupu vyžaduje hodnotu minimální podpory pro získávní frekventovaných množin.

	Apple	Bananas	Beer	Chicken	Milk
0	True	False	True	True	False
1	True	False	False	False	True
2	True	False	True	False	False
3	True	False	True	False	False

Tabulka 4.1: Příklad one-hot kódované tabulky.

	support	itemsets
0	0.8	(Bananas)
1	1.0	(Milk)
2	0.6	(Bananas, Milk)
3	0.6	(Apple, Milk)

Tabulka 4.2: Příklad tabulky s frekventovanými množinami.

Druhá fáze procesu je implementována pomocí metody `association_rules()`. Na vstupu vyžaduje tabulku `DataFrame` se sloupci *support* a *itemsets*, která byla vygenerována pomocí funkce `apriori()` (příklad takové tabulky lze vidět v tabulce 4.2). Na vstupu dále vyžaduje hodnotu minimální spolehlivosti. Výstupem této funkce je tabulka `DataFrame` se sloupci *antecedents* (obsahuje levé strany asociačních pravidel), *consequents* (obsahuje pravé strany asociačních pravidel), *antecedents support* (hodnoty podpory množiny na levé straně asociačních pravidel), *consequents support* (hodnoty podpory množiny na pravé straně asociačních pravidel), *support* (hodnoty podpory asociačních pravidel), *confidence* (hodnoty spolehlivosti asociačních pravidel), atd.

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(Bananas)	(Milk)	1.0	0.8	0.8	0.6	1.0	0.0	1.0
1	(Apple)	(Milk)	0.8	1.0	1.0	0.8	1.0	0.0	inf
2	(Eggs)	(Onions)	0.6	1.0	0.6	0.8	1.0	0.0	inf
3	(Eggs, Bananas)	(Milk)	0.8	0.8	1.0	0.8	1.25	0.12	inf

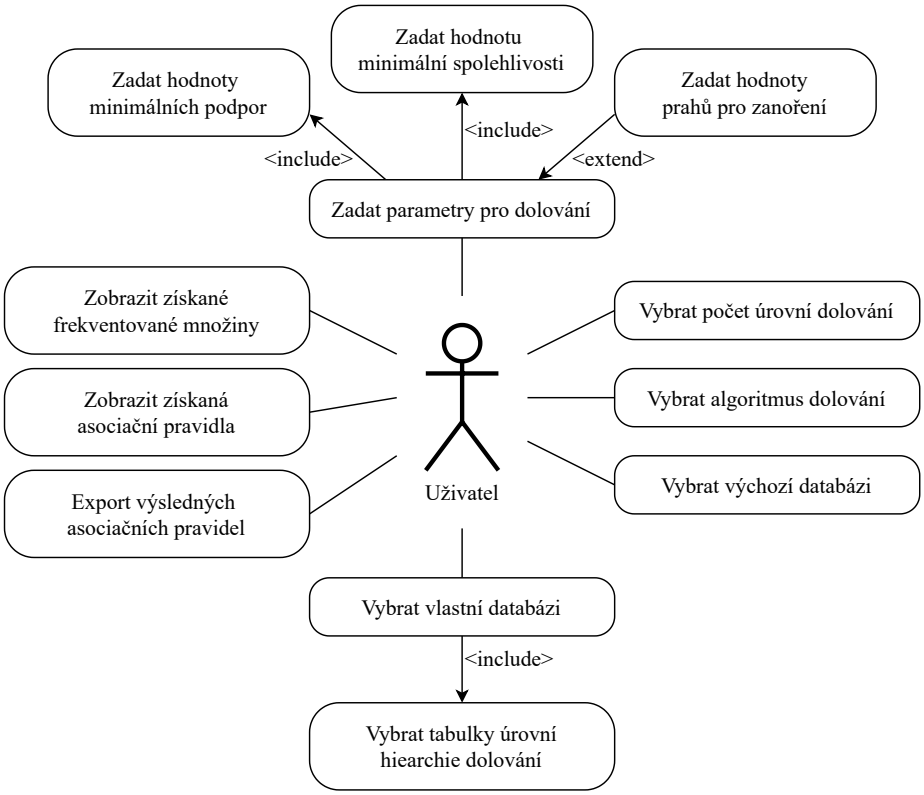
Tabulka 4.3: Příklad tabulky s asociačními pravidly.

4.1.3 Grafické uživatelské rozhraní

Pro grafické uživatelské rozhraní je vybrán framework PyQt verze 5. Tento framework je založený na frameworku Qt. Qt je jedna z nejpopulárnějších multiplatformních knihoven pro vytváření programů s grafickým uživatelským rozhraním. Velkou výhodou této knihovny je možnost použití na všech populárních operačních systémech (jako je např. Windows, Unix, MacOS apod.). Aplikace s grafickým uživatelským rozhraním mají navíc nativní vzhled, který se přizpůsobí danému operačnímu systému. Součástí tohoto frameworku je také nástroj *Qt Designer*, který umožňuje efektivní tvorbu grafických uživatelských rozhraní pomocí způsobu WYSIWYG (what-you-see-is-what-you-get, tzn. co vidíte, to dostanete).

4.1.4 Diagram případů užití

Při práci s aplikací bude existovat pouze jedna role, a to uživatel. Uživatel může vykonávat následující výkony, které jsou vyjádřeny diagramem případů užití níže v 4.1.



Obrázek 4.1: Diagram případů užití.

4.2 Implementace aplikace

Aplikace je implementovaná v programovacím jazyce Python a její grafické uživatelské rozhraní je implementované pomocí knihovny `PyQt5`. Pro získávání víceúrovňových asociačních pravidel je použita knihovna `Mlxtend`. Adresářová struktura projektu se zdrojovými soubory vypadá následovně:

```
/
├── app
│   ├── main.py
│   ├── gui.py
│   └── mining.py
└── data
    ├── categories.csv
    ├── subcategories.csv
    ├── products.csv
    └── transactions.csv
```

Všechny soubory aplikace se nachází ve složce `src`. Ve složce `app` se nacházejí všechny zdrojové soubory potřebné ke spuštění aplikace. Soubor `main.py` slouží k inicializaci aplikace a k jejímu spuštění, soubor `gui.py` implementuje grafické uživatelské rozhraní aplikace a soubor `mining.py` implementuje proces získávání asociačních pravidel. Složka `datasets` obsahuje datové sady, na kterých se bude s implementovanými algortimy získávání asociačních pravidel experimentovat.

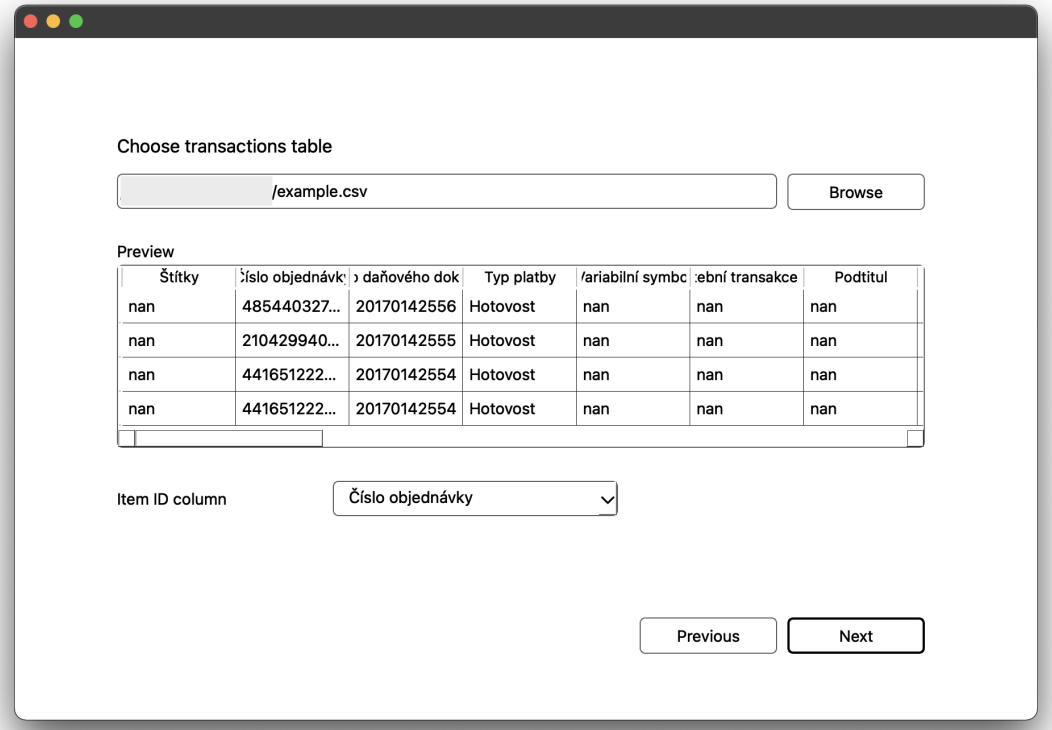
4.2.1 Grafické uživatelské rozhraní

Implementace grafického uživatelského rozhraní se nachází v souboru `gui.py`.

Po spuštění aplikace se objeví hlavní obrazovka reprezentována třídou `Main`. Aplikace obsahuje pouze jedno hlavní okno, ve kterém se bude měnit obsah. To je implementováno pomocí `QStackedWidget`.

Na hlavní obrazovce se vytvoří úvodní stránka, která je reprezentována třídou `StartPage`. Na této stránce má uživatel možnost zvolit si databázi, se kterou budou program pracovat. Na výběr má uživatel výchozí databázi (soubory této databáze se nachází ve složce *datasets*) nebo má uživatel možnost nahrát si vlastní databázi. Pokud uživatel zvolí možnost nahrání vlastní databáze, musí navíc specifikovat počet úrovní konceptuální hierarchie, na kterých bude probíhat dolování asociačních pravidel. Na výběr jsou uživateli celkem tři možnosti, a to počet úrovní roven jedné, dvou nebo tří.

Tlačítkem *Start* se vytvoří další stránky metodou `create_pages()`. Pokud uživatel zvolil možnost nahrání vlastní databáze, vytvoří se stránky, které uživatele provedou nahráváním jednotlivých tabulek. Jedná se o stránky reprezentované třídami `Level1Page`, `Level2Page`, `Level3Page` a `TransPage` a je po uživateli vyžadováno, aby specifikoval jednotlivé tabulky s položkami na odpovídající úrovni a tabulku transakcí. Dále má uživatel za úkol zvolit u každé tabulky sloupec s identifikátory položek a sloupec s popisky položek. Uživateli se přitom navíc zobrazí náhled tabulky, a to pouze první čtyři řádky pro lepší čitelnost, který uživateli usnadní zvolení správné tabulky a správný výběr sloupců. Tlačítkem *Next* u poslední stránky se přejde na stránku, kde se nastavují parametry dolování. Předtím dojde ke kontrole, zda byly zvoleny všechny tabulky a sloupce. Pokud nebyly, zobrazí se dialog s upozorňující zprávou a na další stránku aplikace uživatele pustí až po úspěšném zvolení všech tabulek.

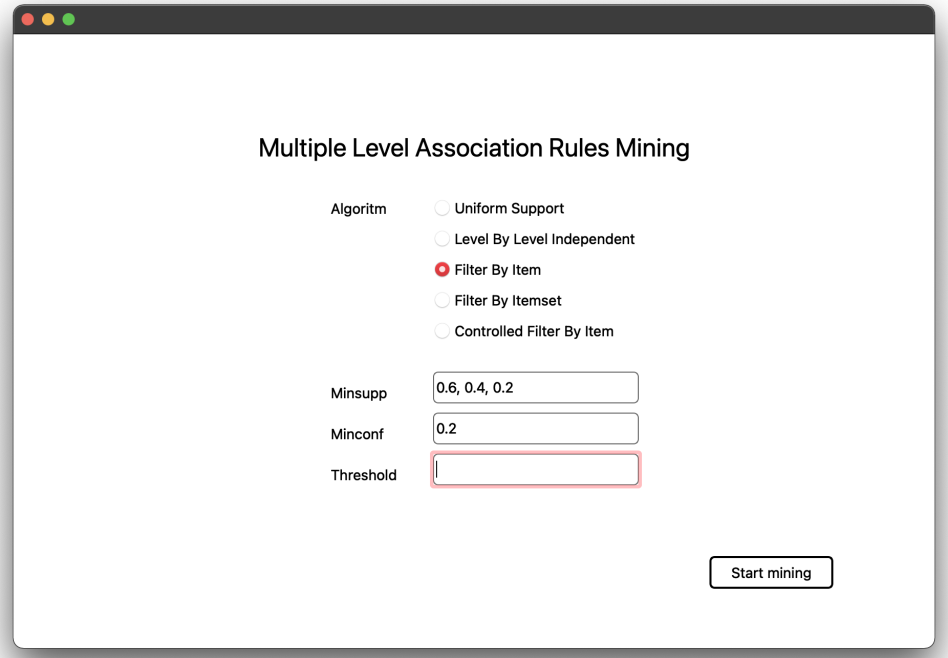


Obrázek 4.2: Výběr vlastní databáze – výběr tabulky transakcí.

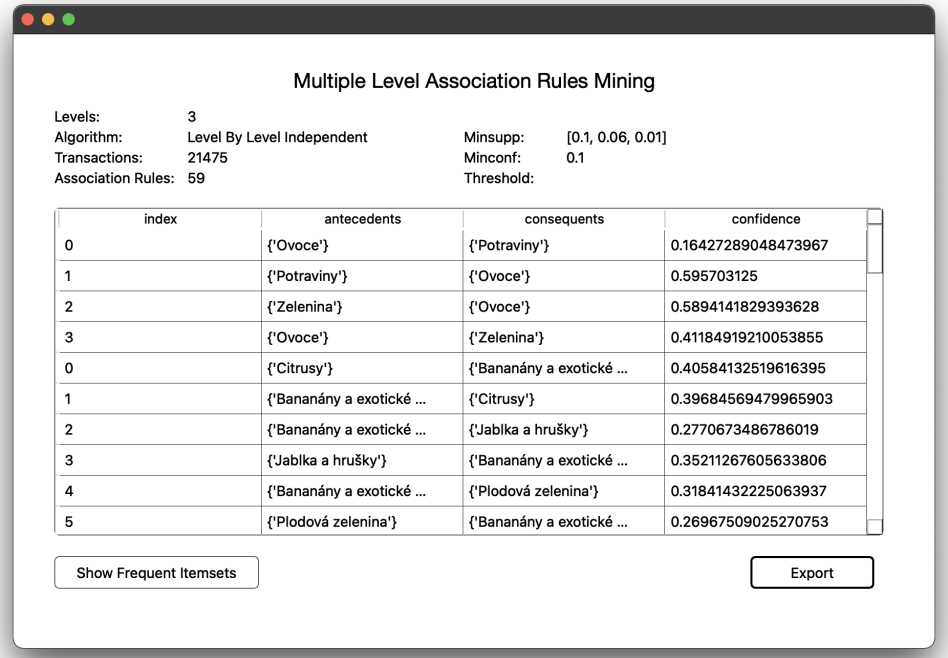
Pokud uživatel nezvolil možnost nahrání vlastní databáze, ale vybral výchozí databázi, je přímo přesměrován na stránku s výběrem parametrů. Tato stránka je reprezentovaná třídou `ConfigPage`. Parametry, které je zde třeba nastavit, jsou hodnoty minimálních podpor a hodnota minimální spolehlivosti. Uživatel na této stránce musí navíc zvolit algoritmus, pomocí kterého proběhne proces dolování. V případě zvolení algoritmu *Kontrolované filtrování úrovní jednou položkou* je navíc potřeba nastavit hodnoty prahů pro zanoření.

K navigaci mezi stránkami slouží tlačítka *Previous* (přejít na předchozí stránku) a *Next* (přejít na další stránku).

Tlačítkem *Start mining* se spustí proces dolování. Předtím dojde ke kontrole správnosti zadaných parametrů (např. zda jsou hodnoty minimálních podpor klesající, zda jsou hodnoty ve správném tvaru, atd.). Po úspěšném nastavení parametrů se vytvoří poslední stránka reprezentovaná třídou `ResultPage`, kde jsou vypsána výsledná získaná asociační pravidla a také přehledně informace o dolování, jako je počet úrovní, zvolený algoritmus, nastavé hodnoty minimální podpory, spolehlivosti a prahů pro zanoření, celkový počet transakcí a celkový počet získaných asociačních pravidel. Tato výsledná asociační pravidla lze také exportovat do souboru JSON, a to pomocí tlačítka *Export*. Aplikace uživateli navíc umožňuje prohlédnout si kromě asociačních pravidel i vygenerované frekventované množiny, a to tlačítkem *Show Frequent Itemsets*.



Obrázek 4.3: Výběr algoritmu a parametrů dolování.



Obrázek 4.4: Zobrazení výsledných získaných asociačních pravidel.

4.2.2 Ukládání databáze

Databáze je ukládána v objektu reprezentovaném třídou `Data`. Pokud byla zvolena možnost nahrání vlastní databáze uživatelem, dojde k zavolání metody `set_custom_database()`, v opačném případě dojde k zavolání metody `set_default_database()`. Jednotlivé tabulky jsou ukládány do struktury `DataFrame` z knihovny *pandas* pomocí metod `get_level1_data()`, `get_level2_data()`, `get_level3_data()` a `get_transactions()`. Potřebné sloupce k procesu dolování se nastavují pomocí metod `set_columns_level1()`, `set_columns_level2()`, `set_columns_level3()` a `set_columns_transactions()`. Tyto sloupce využívá další metoda v této třídě, a to metoda `merge_tables()`, která má za úkol sloučit všechny tabulky do jedné velké transakční tabulky.

4.2.3 Parametry dolování

Pro ukládání uživatelem zadané parametry slouží objekt reprezentován třídou `Config`. Pomocí metody `set_algorithm()` se nastaví zvolený algoritmus. Metoda `set_levels()` nastavuje počet zvolených úrovní, na kterých proběhne dolování asociačních pravidel. Metody `set_minsupp()` a `set_minconf()` ukládají hodnoty minimálních podpor a minimální spolehlivosti, a metoda `set_threshold()` v případě zvolení algoritmu *Kontrolované filtrování úrovní jednou položkou* nastavuje hodnoty prahů pro zanoření do nižší úrovně.

4.2.4 Proces dolování

Aplikace umožňuje získávat víceúrovňová asociační pravidla pomocí pěti různých algoritmů, kde každý algoritmus je reprezentován vlastní třídou. Jedná se tedy o třídy `UniformSupport` (Konstatní podpora), `LevelIndependent` (Redukovaná podpora nezávisle na úrovních), `FilterItem` (Filtrování úrovní jednou položkou), `FilterItemset` (Filtrování úrovní k-množinami) a `ControlledFilterItem` (Kontrolované filtrování úrovní jednou položkou). Tyto třídy dědí od třídy `Mining`, která obsahuje metody `get_all_datasets()`, kterou využívají třídy `UniformSupport` a `LevelIndependent`, a `get_dataset()`, kterou využívají třídy `FilterItem`, `FilterItemset` a `ControlledFilterItem`.

Zavoláním metody `start_mining()` ve třídě `Main` se spustí proces dolování. Na základě uživatelem zvoleného algoritmu se vytvoří objekt příslušné třídy. Nejdříve dojde k načtení tabulek a parametrů zadané uživatelem. Poté dojde ke sloučení všech tabulek do jedné pomocí metody `merge_tables()` třídy `Data`.

Prvním krokem získávání víceúrovňových asociačních pravidel je získávání frekventovaných množin na každé úrovni. To bude probíhat pomocí algoritmu Apriori, který je implementován pomocí metody `apriori()` z knihovny *Mlxtend*. Tyto frekventované množiny budou sloužit jako vstup metodě `association_rules()` ze stejné knihovny, která implementuje generování výsledných asociačních pravidel.

U všech algoritmů dojde nejdříve k transformování všech transakčních tabulek na všech úrovních na tabulky v one-hot kódování, a to pomocí metody `get_all_datasets()` z rodičovské třídy `Mining`. Poté se na každé úrovni aplikuje algoritmus Apriori s uživatelem specifikovanou hodnotou minimální podpory metodou `apriori()` (jinými slovy se z transakčních tabulek na každé úrovni vygenerují frekventované množiny).

Algoritmy *Konstantní podpora* (třída `UniformSupport`) a *Nezávisle na úrovních* (třída `LevelIndependent` pracují na stejném principu. Jediný rozdíl u nich je, že u algoritmu *Konstantní podpora* má každá úroveň hierarchie stejnou hodnotu minimální podpory a oproti tomu algoritmus *Nezávisle na úrovních* má na každé úrovni jinou hodnotu minimální pod-

pory. Dále se z těchto frekventovaných množin vygenerují výsledná asociační pravidla s uživatelem zadanou hodnotou minimální spolehlivosti metodou `association_rules()`.

U algoritmů *Filtrování úrovně položkou*, *Filtrování úrovně k-množinou* a *Kontrolované filtrování úrovně položkou* (třídy `FilterItem`, `FilterItemset`, `ControlledFilterItem`) po vygenerování frekventovaných množin dojde navíc k filtrování položek. Pouze na nejvyšší úrovni nedochází k žádnému filtrování nedochází, jinak na vyšších úrovních dochází k vytváření filtru pomocí metody `get_filter()` a podle tohoto filtru dojde na nižších úrovních k odstranění položek, které se ve filtru nenachází, a to pomocí metody `delete_items()`. Po úspěšném vyfiltrování frekventovaných množin se pomocí metody `association_rules()` z těchto frekventovaných množin vygenerují výsledná asociační pravidla.

Algoritmus *Kontrolované filtrování úrovně položkou* k vytváření frekventovaných množin využívá hodnotu prahu pro zanoření, ale do výsledných frekventovaných množin se vyberou pouze ty množiny, které splňují podmínku minimální podpory. Filtr se u tohoto algoritmu vytváří z frekventovaných množin získaných pomocí hodnot prahů pro zanoření. Výsledná asociační pravidla jsou vygenerována pouze z výsledných frekventovaných množin splňující podmínku minimální podpory.

Výsledná získaná asociační pravidla vypsána zavoláním metody `show_result()` třídy `Main` v podobě tabulky `DataFrame` se sloupci *antecedents*, *consequents* a *confidence*. Metodou `show_frequent()` jsou vypsány získané frekventované množiny také v podobě tabulky se sloupci *itemsets* a *support*. Pomocí metody `export_rules()` lze tato asociační pravidla exportovat do souboru JSON.

Kapitola 5

Experimenty a vyhodnocení

Tato kapitola popisuje experimenty vykonávané s algoritmy pomocí implementované aplikace, které byly popsané výše v teoretické části v sekci 3.6.1. Podkapitola 5.1 popisuje poskytnutou databázi, na které budou experimenty probíhat. Podkapitola 5.2 popisuje provedené experimenty pomocí jednotlivých algoritmů a v podkapitole 5.3 jsou tyto výsledky porovnány a zhodnoceny.

5.1 Databáze

Databáze, na které budou probíhat experimenty, se skládá z celkem tří tabulek, a to *transactions*, *categories* a *products*. Z toho tedy vyplývá, že databáze poskytuje položky rozdělené pouze do dvou úrovních konceptuální hierarchie. Pro účely této aplikace a otestování získávání asociačních pravidel na více úrovních jsem do databáze vytvořila jednu novou tabulku, která navíc obsahuje podkategorie produktů. Nyní jsou tedy položky rozděleny do tří úrovní konceptuální hierarchie. Jelikož se jedná o reálná data, jsou na požadavky majitele maloobchodu z databáze odstraněny sloupce tabulek, které nejsou pro účel získávání asociačních pravidel potřebné (jako např. cena, čas transakce, hmotnost produktu, barva při zobrazení v pokladním systému, způsob platby transakce, apod.). Nynější podoba databáze se tedy skládá z následujících tabulek:

- *transactions* – jedná se o tabulku transakcí, obsahuje sloupce *transactionId* (identifikátor transakce) a *productId* (identifikátor produktu)
- *categories* – jedná se to tabulku kategorií produktů, obsahuje sloupce *categoryId* (identifikátor kategorie) a *categoryName* (název kategorie)
- *subcategories* – jedná se o tabulku podkategorií produktů, obsahuje sloupce *subcategoryId* (identifikátor podkategorie) a *subcategoryName* (název podkategorie)
- *products* – jedná se o tabulku produktů, obsahuje sloupce *productId* (identifikátor produktu), *categoryId* (identifikátor kategorie, do které produkt patří) a *subcategoryId* (identifikátor podkategorie, do které produkt patří)

Databáze má následující vlastnosti:

- Databáze obsahuje 845 produktů
- Produkty jsou rozděleny do 3 úrovní konceptuální hierarchie

- Databáze obsahuje 40 podkategorií
- Databáze obsahuje 6 kategorií
- Databáze obsahuje celkem 21475 transakcí (22550 záznamů)

5.2 Experimenty

Tato podkapitola se zaměřuje na experimenty s databází popsanou výše v 5.1 pomocí pěti algoritmů, které jsou popsané v teoretické části práce (3.6.1). Celkem jsou provedeny tři experimenty. První experiment 5.2.1 je vykonán pomocí algoritmu *Konstantní podpora* a experimenty 5.2.2 a 5.2.3 pracují se zbylými čtyřmi algoritmy (*Nezávisle na úrovních*, *Filtrování úrovně položkou*, *Filtrování úrovně k-množinou* a *Kontrolované filtrování úrovně položkou*).

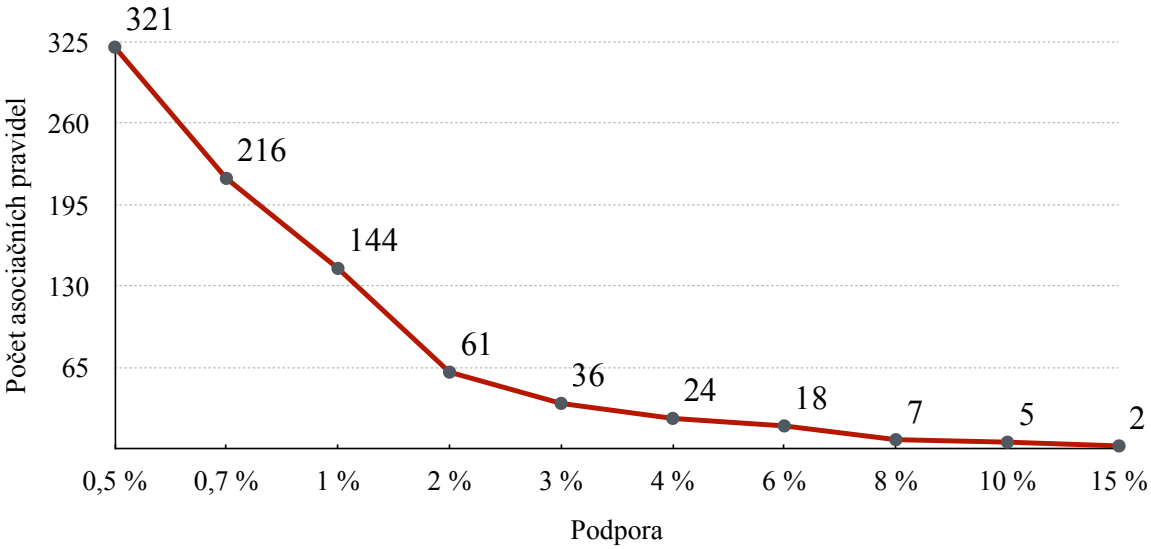
5.2.1 Experiment 1

V prvním experimentu jsou pro všechny úrovně nastaveny stejné hodnoty minimální podpory. Jedná se tedy o algoritmus s **konstantní podporou**. Hodnota minimální spolehlivosti je na všech úrovních stejná a je nastavena na 20%. Výsledky tohoto experimentu popisuje tabulka 5.1 a jsou také graficky znázorněny na grafu 5.1.

Podpora	**Počet pravidel na 1. úrovni**	**Počet pravidel na 2. úrovni**	**Počet pravidel na 3. úrovni**	**Počet pravidel celkem**
15%	2	0	0	2
10%	3	2	0	5
8%	3	4	0	7
6%	4	12	2	18
4%	7	15	2	24
3%	9	25	2	36
2%	12	40	9	61
1%	24	92	28	144
0,7%	38	122	56	216
0,5%	51	195	75	321

Tabulka 5.1: Experiment 1 – Počet nalezených asociačních pravidel na jednotlivých úrovních hierarchie.

Z tohoto experimentu je zjištěno, že na první úrovni hierarchie je vhodné pracovat s hodnotou minimální podpory menší než 30%, jelikož s vyšší hodnotou minimální podpory nebyl algoritmus schopen vygenerovat žádná asociační pravidla. S hodnotou podpory mezi 30% až 11% vygeneroval algoritmus shodná asociační pravidla. Jak vyplývá z výsledků, na druhé úrovni hierarchie je vhodné pracovat s hodnotou podpory menší než 10% a na třetí úrovni s hodnotou menší než 6%.



Obrázek 5.1: Závislost počtu nalezených asociačních pravidel na hodnotě podpory.

5.2.2 Experiment 2

Úkolem tohoto experimentu je získat asociační pravila pomocí každého z implementovaných algoritmů, a to se stejnými parametry. Parametry jsou zvoleny na základě předchozího experimentu (5.2.1) a jsou následující:

- hodnoty minimálních podpor: 15%, 7%, 1%
- hodnota minimální spolehlivosti: 20%
- hodnoty prahů pro zanoření: $11,5\%$, $3,5\%$ (pouze u algoritmu *Kontrolované filtrování úrovně jednou položkou*)

V tomto experimentu již není prováděno získávání asociačních pravidel pomocí algoritmu *Konstantní podpora*, jelikož výsledky tohoto algoritmu lze vidět v předchozím experimentu.

	Podpora $= 15\%, 7\%, 1\%$
Počet pravidel na 1. úrovni	2
Počet pravidel na 2. úrovni	8
Počet pravidel na 3. úrovni	28
Počet pravidel celkem	38

Tabulka 5.2: Výsledky algoritmu *Nezávisle na úrovních*.

	Podpora $= 15\%, 7\%, 1\%$
Počet pravidel na 1. úrovni	2
Počet pravidel na 2. úrovni	8
Počet pravidel na 3. úrovni	26
Počet pravidel celkem	36

Tabulka 5.3: Výsledky algoritmu *Filtrování úrovně položkou.*

	Podpora $= 15\%, 7\%, 1\%$
Počet pravidel na 1. úrovni	2
Počet pravidel na 2. úrovni	8
Počet pravidel na 3. úrovni	20
Počet pravidel celkem	30

Tabulka 5.4: Výsledky algoritmu *Filtrování úrovně k-množinou.*

	Podpora $= 15\%, 7\%, 1\%$
Počet pravidel na 1. úrovni	2
Počet pravidel na 2. úrovni	8
Počet pravidel na 3. úrovni	28
Počet pravidel celkem	38

Tabulka 5.5: Výsledky algoritmu *Kontrolované filtrování úrovně položkou.*

V tabulkách výše (5.2, 5.3, 5.4, 5.5) je možné vidět získaná asociační pravidla pomocí jednotlivých algoritmů, a to i počet pravidel na jednotlivých úrovních hierarchie. Nejmenšího počtu asociačních pravidel dosáhl algoritmus *Filtrování úrovně k-množinou* a největšího počtu asociančních pravidel dosáhy algoritmy *Nezávisle na úrovních* a *Kontrolované filtrování úrovně položkou.*

V tabulkách 5.6 a 5.7 níže jsou shrnuty výsledky implementovaných algoritmů.

	Počet pravidel
Nezávisle na úrovních	38
Filtrování úrovně položkou	36
Filtrování úrovně k-množinou	30
Kontrolované filtrování úrovně položkou	38

Tabulka 5.6: Výsledky implementovaných algoritmů – asociační pravidla.

	Frekv. mn. na 1. úrovni	Frekv. mn. na 2. úrovni	Frekv. mn. na 3. úrovni	Frekv. mn. celkem
Nezávisle na úrovních	4	13	74	91
Filtrování úrovně položkou	4	12	63	79
Filtrování úrovně k-množinou	4	11	45	60
Kontrolované filtrování úrovně položkou	4	13	71	88

Tabulka 5.7: Výsledky implementovaných algoritmů – frekventované množiny.

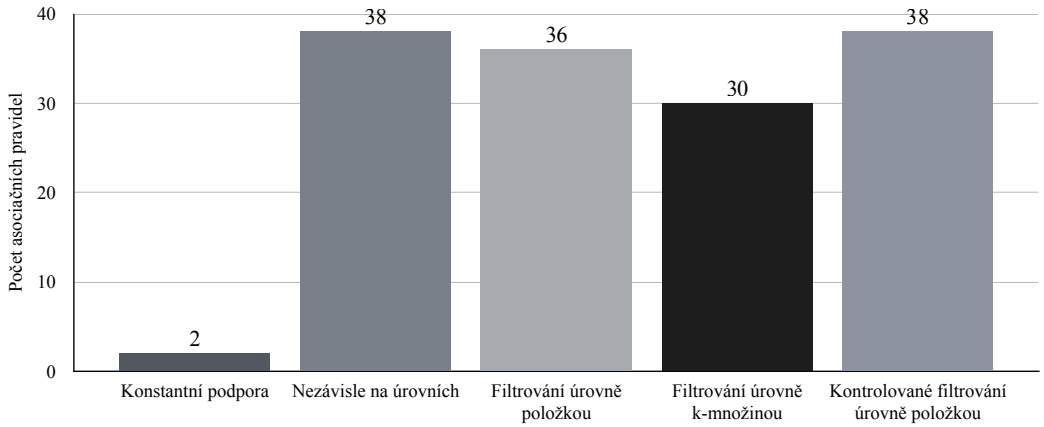
Získané frekventované množiny jsou u všech algoritmů na první úrovni hierarchie identické, protože zde nedochází u žádného algoritmu k žádnému filtrování, proto jsou tedy i získaná asociační pravidla u všech algoritmů na první úrovni stejná.

Na druhé úrovni hierarchie dosáhl nejmenšího počtu frekventovaných množin algoritmus *Filtrování úrovně k-množinou* a největšího počtu frekventovaných množin dosáhl algoritmus *Nezávisle na úrovních*. U algoritmu *Filtrování úrovně položkou* došlo k odstranění položky *{Cigarety}*, protože její rodičovká položka nebyla frekventovaná. U algoritmu *Filtrování úrovně k-množinou* také došlo k odstranění položek, a to *{Cigarety}* a *{Sladkosti}*, protože jejich rodičovské položky nepatřily do množiny frekventovaných k-množin.

Na třetí úrovni hierarchie je nejmenšího počtu frekventovaných množin opět u algoritmu *Filtrování úrovně k-množinou* a největšího počtu opět u algoritmu *Nezávisle na úrovních*.

Příklad asociačního pravidla, který algoritmus *Filtrování úrovně položkou* neodhalil, ale nachází se v asociačních pravidlech algoritmu *Nezávisle na úrovních*, je například pravidlo *Mrkev* $\Rightarrow$ *Petržel* $[c = 47,619\%]$. Toto asociační pravidlo se zde nenachází, protože rodičovská položka těchto položek (*{Mrkev a kořenová zelenina}*) nebyla na vyšší úrovni frekventovaná. Oproti tomu se toto asociační pravidlo nachází u algoritmu *Kontrolované filtrování úrovně položkou* díky hodnotě prahu pro zanoření.

Příkladem asociačního pravidla, který algoritmus *Filtrování úrovně k-množinou* nebyl schopen získat, ale nachází se u ostatních algoritmů (kromě algoritmu *Konstantní podpora*), je například pravidlo *Hrozny bílé* $\wedge$ *Banány* $\Rightarrow$ *Mandarinky 1* $[c = 38,3673\%]$, protože rodičovská položka (*{Hrozny a peckoviny}*) na vyšší úrovni nepatřila do množiny frekventovaných k-množin.



Obrázek 5.2: Výsledky všech implementovaných algoritmů.

5.2.3 Experiment 3

Úkolem tohoto experimentu je stejný jako u experimentu 5.2.2, ale jsou zde nastaveny nižší hodnoty parametrů. Jsou tedy následující:

- hodnoty minimálních podpor: 8%; 4,1%; 0,2%
- hodnota minimální spolehlivosti: 20%
- hodnoty prahů pro zanoření: 6%; 2% (pouze u algoritmu *Kontrolované filtrování úrovně jednou položkou*)

	Podpora = 8%; 4,1%; 0,2%
Počet pravidel na 1. úrovni	3
Počet pravidel na 2. úrovni	14
Počet pravidel na 3. úrovni	246
Počet pravidel celkem	263

Tabulka 5.8: Výsledky algoritmu *Nezávisle na úrovních*.

	Podpora = 8%; 4,1%; 0,2%
Počet pravidel na 1. úrovni	3
Počet pravidel na 2. úrovni	14
Počet pravidel na 3. úrovni	235
Počet pravidel celkem	252

Tabulka 5.9: Výsledky algoritmu *Filtrování úrovně položkou*.

	Podpora = 8%; 4,1%; 0,2%
Počet pravidel na 1. úrovni	3
Počet pravidel na 2. úrovni	14
Počet pravidel na 3. úrovni	169
Počet pravidel celkem	186

Tabulka 5.10: Výsledky algoritmu *Filtrování úrovně k-množinou*.

	Podpora $= 8\%; 4,1\%; 0,2\%$
Počet pravidel na 1. úrovni	3
Počet pravidel na 2. úrovni	14
Počet pravidel na 3. úrovni	244
Počet pravidel celkem	261

Tabulka 5.11: Výsledky algoritmu *Kontrolované filtrování úrovně položkou.*

Z tabulek výsledků (5.8, 5.9, 5.10, 5.11) vyplývá, že největšího počtu nalezených asociačních pravidel dosáhl algoritmus *Nezávisle na úrovních* a nejmenšího počtu asociačních pravidel dosáhl algoritmus *Filtrování úrovně k-množinou.*

V tabulkách 5.12 a 5.13 níže jsou shrnuty výsledky implementovaných algoritmů.

	Počet pravidel
Nezávisle na úrovních	263
Filtrování úrovně položkou	252
Filtrování úrovně k-množinou	186
Kontrolované filtrování úrovně položkou	261

Tabulka 5.12: Výsledky implementovaných algoritmů.

	Frekv. mn. na 1. úrovni	**Frekv. mn. na 2. úrovni**	**Frekv. mn. na 3. úrovni**	**Frekv. mn. celkem**
Nezávisle na úrovních	6	20	491	517
Filtrování úrovně položkou	6	19	441	466
Filtrování úrovně k-množinou	6	18	283	307
Kontrolované filtrování úrovně položkou	6	20	468	494

Tabulka 5.13: Výsledky implementovaných algoritmů – frekventované množiny.

Na první úrovni hiearchie jsou nalezené frekventované množiny stejné u všech algoritmů, protože zde nedochází k žádnému filtrování. Tím pádem jsou i asociační pravidla získaná na této úrovni stejné.

Na druhé úrovni hierarchie je nejmenší počet frekventovaných množin u algoritmu *Filtrování úrovně k-množinou* a největší počet je u algoritmů *Nezávisle na úrovních* a *Kontrolované filtrování úrovně položkou.* I přesto je počet získaných asociačních pravidel na této úrovni u všech algoritmů stejný.

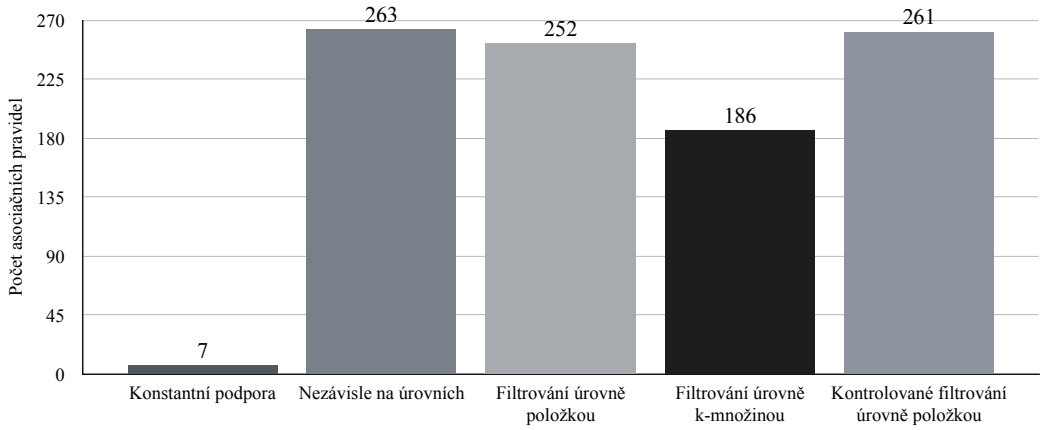
K rozdílu v počtu získaných asociačních pravidel dochází až na třetí úrovni hierarchie. Nejmenšího počtu frekventovaných množin dosáhl algoritmus *Filtrování úrovně k-množinou* a největšího počtu dosáhl algoritmus *Nezávisle na úrovních*, stejně jako na vyšší úrovni.

Příklad asociačního pravidla, který algoritmus *Filtrování úrovně položkou* neodhalil, ale nachází se v pravidlech algoritmu *Nezávisle na úrovních*, je například pravidlo *Kozel 11° 0,5l plech* $\Rightarrow$ *Camel filters* $[c = 75,3086\%]$. Tento algoritmus toto pravidlo neobsahuje,

protože rodičovská položka položky *Kozel 11° 0,5l plech* (*Pivo*) nebyla na vyšší úrovni frekventovaná, zatímco algoritmus *Nezávisle na úrovních* testuje i položky, u kterých je rodičovská položka nefrekventovaná.

Algoritmus *Kontrolované filtrování úrovně položkou* obsahuje oproti algoritmu *Filtrování úrovně položkou* o 9 asociačních pravidel více. Jedním z nich je například pravidlo *Salat hlavkovy* $\Rightarrow$ *Banány* $[c = 37,1795\%]$. Položka *Salat hlavkovy* se u algoritmu *Filtrování úrovně položkou* nenachází, protože její rodičovská položka (*{Listová zelenina a saláty}* nebyla na vyšší úrovni frekventovaná, ale u algoritmu *Kontrolované filtrování úrovně položkou* odstraněna není, protože splňuje hodnotu prahu pro zanoření.

Příkladem asociačního pravidla, který algoritmus *Filtrování úrovně k-množinou* nebyl schopen získat, ale nachází se u ostatních algoritmů (kromě algoritmu *Konstantní podpora*), je například pravidlo *Banány* $\wedge$ *Brambory rané* $\Rightarrow$ *Mandarinky 1* $[c = 33,8235\%]$, protože rodičovská položka (*{Brambory a cibuloviny}*) na vyšší úrovni nepatřila do množiny frekventovaných k-množin.



Obrázek 5.3: Výsledky všech implementovaných algoritmů.

5.3 Zhodnocení experimentů

Na základě výsledků experimentů lze vidět, že nejméně asociačních pravidel vygeneroval algoritmus *Konstantní podpora*, který v experimentu 5.2.3 dokázal získat pouze 7 pravidel oproti zbylým algoritmům, které v průměru získaly 240 pravidel. Z toho lze říci, že tento algoritmus není příliš vhodný pro získávání pravidel na více úrovních. Protože se položky na nejnižší úrovni nevyskytují tak často jako položky na vyšších úrovních, při zvolení příliš vysoké podpory dochází ke ztrátě pravidel na nižších úrovních a naopak, při zvolení příliš nízké hodnoty dochází k získávání asociací mezi téměř všemi položkami na vyšších úrovních, tím pádem nejsou výsledkem příliš zajímavá asociační pravidla.

Dále u algoritmu *Filtrování úrovně k-množinou* lze říci, že kvůli příliš silnému omezení také dochází ke ztrátě některých zajímavých pravidel. Například u tohoto algoritmu v experimentu 5.2.2 není nalezeno pravidlo $\{Mandarinky\ 1\} \wedge \{Hrozny\ bílé\} \Rightarrow \{Banány\}$, u kterého je spolehlivost až $44,13\%$. Jednalo by se o druhé pravidlo s nejvyšší hodnotou spolehlivostí (pravidlo s nejvyšší hodnotou spolehlivostí je zde pravidlo $\{Zelenina\} \Rightarrow \{Ovoce\}$ $[c = 58,94\%]$), ale kvůli filtrování bylo odstraněno.

Další tři algoritmy *Nezávisle na úrovních*, *Filtrování úrovně položkou* a *Kontrolované filtrování položkou* se od sebe počtem získaných asociačních pravidel příliš neliší, ale s větším objemem dat by mohl být rozdíl patrnější. Z experimentů se také ukázalo, že z těchto tří algoritmů byl algoritmus *Filtrování úrovně položkou* nejrychlejší, a algoritmy *Nezávisle na úrovních* a *Kontrolované filtrování položkou* se od sebe časově téměř nelišily. Asociační pravidla s nejvyšší hodnotou spolehlivosti, které se nachází u těchto třech algoritmů, jsou například:

- $Mrkev \wedge Celer,\ Banány \Rightarrow Petržel\ [c = 90\%]$
- $Petržel \wedge Cibule \Rightarrow Mrkev\ [c = 86,36\%]$
- $Petržel \wedge Brambory\ rané \Rightarrow Mrkev\ [c = 81,82\%]$
- $Petržel \wedge Celer \wedge Banány \Rightarrow Mrkev\ [c = 78,26\%]$
- $Petržel \wedge Citron \Rightarrow Mrkev\ [c = 75,86\%]$

Kapitola 6

Závěr

Tato bakalářská práce se zabývá problematikou získávání asociačních pravidel, a to konkrétně víceúrovňových. Práce se zaměřuje na metody získávání víceúrovňových asociačních pravidel a cílem bylo tyto zvolené metody experimentálně porovnat. Dalším úkolem práce bylo implementovat aplikaci s jednoduchým grafickým uživatelským rozhraním, pomocí které bude probíhat získávání asociačních pravidel. Byly zvoleny algoritmy založené na agloritmu Apriori, a to konkrétně algoritmy *Konstantní podpora*, *Nezávisle na úrovních*, *Filtrování úrovně položkou*, *Filtrování úrovně k-množinou* a *Kontrolované filtrování úrovně položkou*.

První část práce se věnuje problematice získávání znalostí z databází a získávání asociačních pravidel, podrobněji se zaměřuje na víceúrovňová asociačních pravidla. V další části práce je popsán návrh a implementace experimentální aplikace. Poslední část práce se zaměřuje na experimenty s implementovanými algoritmy pomocí implementované aplikace a na závěr byly výsledky algoritmů mezi sebou porovnány.

Práci je dále možno rozšířit o další alternativní algoritmy pro získávání víceúrovňových asociačních pravidel, jako například o algoritmy z rodiny *ML* publikované autory Jiawei Han a Yongjian Fu. Dalším rozšířením aplikace by mohla být možnost dolování na více než třech úrovních konceptuální hiearchie.

Literatura

[1] Dunham, M. H. *Data Mining: Introductory and Advanced Topics*. 1. vyd. Prentice-Hall, 2003. ISBN 0-13-088892-3.

[2] Fayyad, U., Piatetsky Shapiro, G. a Smyth, P. From Data Mining to Knowledge Discovery in Databases. *AI Magazine*. 1996, Volume 17, č. 3.

[3] Han, J. a Fu, Y. Discovery of Multiple-Level Association Rules from Large Databases. *VLDB*. 1999-09, sv. 11, č. 11.

[4] Han, J., Kamber, M. a Pei, J. *Data Mining: Concepts and Techniques*. 3. vyd. Morgan Kaufmann Publishers, 2012. 703 s. ISBN 978-0-12-381479-1.

[5] Pecinovský, R. *Začínáme programovat v jazyku Python*. 1. vyd. Grada Publishing, 2020. ISBN 978-80-271-1237-1.

[6] Raschka, S. MLxtend: Providing machine learning and data science utilities and extensions to Python's scientific computing stack. *The Journal of Open Source Software*. The Open Journal. duben 2018, sv. 3, č. 24. DOI: 10.21105/joss.00638. Dostupné z: http://joss.theoj.org/papers/10.21105/joss.00638.

[7] Tan, P.-N., Steinbach, M. a Kumar, V. *Introduction to Data Mining*. 1. vyd. Pearson Education, Inc., 2006. 769 s. ISBN 0-321-42052-7.

[8] Zendulka, J., Bartík, V., Roman, L. a Rudolfová, I. *Získávání znalostí z databází. Studijní opora*. Vysoké učení technické v Brně, Fakulta informačních technologií, 2006.

Příloha A

Obsah přiloženého paměťového média

Součástí této práce je paměťové médium s následujícím obsahem:

- `app` – složka obsahující zdrojové soubory aplikace
- `data` – složka obsahující experimentální databázi
- `tex` – složka obsahující zdrojové soubory bakalářské práce v LaTeX
- `xnguye18.pdf` – bakalářská práce ve formátu `pdf`
- `README.md` – návod na spuštění aplikace