VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

ZÍSKÁVÁNÍ ZNALOSTÍ V PROSTŘEDÍ MS SQL

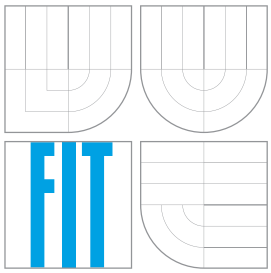
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

ROMAN PIJÁČEK

BRNO 2011

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

ZÍSKÁVÁNÍ ZNALOSTÍ V PROSTŘEDÍ MS SQL

KNOWLEDGE DISCOVERY IN MS SQL ENVIRONMENT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

ROMAN PIJÁČEK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. VLADIMÍR BARTÍK, Ph.D.

BRNO 2011

Zadání bakalářské práce/12378/2010/xpijac00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2010/2011

Zadání bakalářské práce

Řešitel: **Pijáček Roman**
Obor: Informační technologie
Téma: **Získávání znalostí v prostředí MS SQL**
Knowledge Discovery in MS SQL Environment
Kategorie: Data mining

Pokyny:

1. Seznamte se s problematikou získávání znalostí z databází.
2. Prostudujte podporu pro získávání znalostí v prostředí MS SQL Serveru. Po dohodě s vedoucím zvolte metodu získávání znalostí, která v MS SQL podporována není, a tu prostudujte podrobněji.
3. Navrhněte aplikaci, která bude provádět získávání znalostí s využitím metod podporovaných MS SQL a zvolené metody, která podporována není.
4. Navrženou aplikaci implementujte a ověřte její funkčnost.
5. Testujte jednotlivé metody na vhodně zvoleném vzorku dat.
6. Zhodnoťte dosažené výsledky a další možné pokračování v projektu.

Literatura:

- Zendulka, J. a kol.: Získávání znalostí z databází. Studijní opora, FIT VUT, 2006.
- Han, J., Kamber, M.: Data Mining - Concepts and Techniques, 2nd Edition. Morgan Kaufmann Publishers, 2006.

Při obhajobě semestrální části projektu je požadováno:

- Body 1-3

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Bartík Vladimír, Ing., Ph.D.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2010
Datum odevzdání: 18. května 2011

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Bakalářská práce se zabývá problematikou získávání znalostí z databází v prostředí MS SQL Serveru 2008. Po počátečním uvedení do oblasti získávání znalostí z databází jsou detailněji vysvětleny obecné principy a algoritmy použitých dolovacích metod (Bayesovská klasifikace, asociační pravidla, rozhodovací stromy, shluková analýza). V rámci praktické části této práce je navržena, implementována a otestována klientská desktopová aplikace v jazyce C#, která uživateli umožňuje interaktivní získávání klíčových znalostí a skrytých informací z databází. Aplikace využívá metod zabudovaných v rámci MS SQL Serveru 2008 a také metody Apriori pro dolování silných asociačních pravidel, která byla implementována ve vlastní režii. V závěru jsou diskutována další možná rozšíření stávajícího projektu a uvedeno zhodnocení dosažených výsledků.

Abstract

This Bachelor's thesis deals with issue of knowledge discovery in databases in MS SQL Server 2008. After the initial entry into the field of knowledge discovery in databases, general principles and algorithms of used data mining methods (Bayesian classification, association rules, decision trees, cluster analysis) are explained in detail. In the practical part of this thesis there is designed, implemented and tested desktop application which allows the user to discover knowledge and hidden information from database. The application uses methods built into the SQL Server 2008 and the Apriori method for mining strong association rules. In the end, there are discussed further possible expansion of the existing project and an evaluation of the results.

Klíčová slova

Získávání znalostí z databází, dolování z dat, asociační pravidla, Apriori, Bayesovská klasifikace, rozhodovací stromy, shluková analýza, MS SQL Server 2008, T-SQL, C#, DMX, ADOMD.NET, .NET, OOP

Keywords

Knowledge discovery in databases, data mining, association rules, Apriori, Bayesian classification, decision trees, cluster analysis, MS SQL Server 2008, T-SQL, C#, DMX, ADOMD.NET, .NET, OOP

Citace

Roman Pijáček: Získávání znalostí v prostředí MS SQL, bakalářská práce, Brno, FIT VUT v Brně, 2011

Získávání znalostí v prostředí MS SQL

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Vladimíra Bartíka, Ph.D.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Roman Pijáček
7. května 2011

Poděkování

Děkuji panu Ing. Vladimíru Bartíkovi, Ph.D. za profesionální přístup, konstruktivní připomínky a cenné rady při vedení bakalářské práce.

Obsah

Kapitola 1

Úvod

V dnešním světě datové a informační exploze je čím dál tím náročnější data nejenom spolehlivě a bezpečně uchovávat, ale také je umět plně využívat ve svůj prospěch a vydolovat z nich zajímavé a užitečné znalosti. Drtivá většina uznávaných a celosvětově úspěšných společností si již význam získávání znalostí z databází plně uvědomuje a tato oblast se stala nedílnou součástí jejich firemní infrastruktury. Na základě vydolovaných znalostí pak společnosti mohou učinit klíčová obchodní a marketingová rozhodnutí. Obchodní a marketingová rozhodnutí jsou součástí *Business Intelligence*, kterému je věnována pozornost v **kapitole druhé**. Kromě zavedení pojmu *Business Intelligence* jsou v této kapitole představeny i jednotlivé fáze procesu získávání znalostí z databází a nechybí zde ani využití a uplatnění dolování z dat v praxi.

Proces získávání znalostí z databází je poměrně složitý a musíme si předem uvědomit, jaké vzory chceme z dat získat a jakou dolovací úlohu hodláme vlastně řešit. Pro usnadnění volby dolovací úlohy jsou ve **třetí kapitole** hlavní typy dolovacích úloh a algoritmů, včetně jejich principů, představeny.

Tato bakalářská práce se podrobněji zabývá čtyřmi typy dolovacích úloh. Jedná se o nalezení frekventovaných množin a silných asociačních pravidel a dále pak o rozhodovací stromy, Bayesovskou klasifikaci a shlukování. Teoretické základy a principy těchto vybraných úloh jsou charakterizovány v **kapitole čtvrté**.

Nástroje pro dolování z dat bývají zakomponovány buď v některých ERP systémech nebo jsou součástí jiných platforem. Mezi nejznámější a nejpoužívanější nástroje patří například MS SQL Server 2008, databázový server Oracle, SAS Enterprise Miner a další. Pro praktickou část této práce byl zvolen databázový MS SQL Server 2008, jehož moduly jsou krátce charakterizovány v **kapitole páté**.

Šestá kapitola obsahuje specifikaci požadavků na aplikaci, která bude umožňovat uživatelům získávání znalostí z databází, a to jednak s využitím metod zabudovaných v rámci MS SQL Serveru a také s využitím vlastní implementované metody. V šesté kapitole je čtenář seznámen s technologiemi, na kterých je aplikace postavena a s návrhem serverové a klientské části. Návrh klientské části obsahuje i vysvětlení architektury aplikace.

Implementaci celého projektu shruje **kapitola sedmá**, která je dělena na dvě části. První část si klade za cíl přiblížit implementaci dolovacích modelů na straně serveru a smyslem druhé části je shrnout průběh implementace klientské desktopové aplikace. V implementaci klienta se lze dočíst o jednotlivých vrstvách, ale také o práci s vlákny, nebo o dalších funkcionalitách, kterými implementovaná aplikace Data Miner 2011 disponuje.

Nedílnou součástí životního cyklu softwaru je bezesporu i zkoumání jeho korektního chování a jeho testování, kterým se zabývá **kapitola osmá**. Kromě seznámení se s průběhem

testování aplikace může čtenář zjistit, jaké funkcionality byly na základě připomínek ze strany uživatelů do aplikace dodatečně implementovány a jaká jsou navrhovaná další možná rozšíření projektu do budoucna.

Závěrečná **devátá kapitola** shrnuje celkový přínos bakalářské práce a hodnotí výsledky, kterých bylo dosaženo. Dále také obsahuje zhodnocení dalšího možného vývoje projektu.

Kapitola 2

Získávání znalostí z databází

V dnešním světě informačních technologií prudce roste objem dat uchovávaných v databázích. Moderní databázové servery nám umožňují nejen rychlou, ale také bezpečnou práci s takovými daty. S velikostí ukládaných dat však narůstá i potřeba získávat z dat užitečné informace, či znalosti. Pojmy data a informace bývají často zaměňovány nebo dokonce chápány jako pojmy identické. Není tomu tak, poněvadž data se stávají informacemi, pokud:

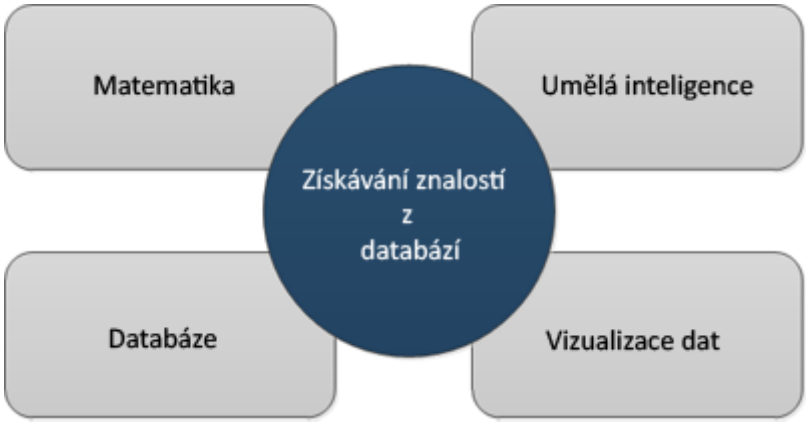
- máte data,
- víte, že máte data,
- víte, kde tato data máte uložena,
- máte k datům přístup,
- zdroji dat můžete důvěřovat,
- uložená data mají význam.

Historie získávání znalostí z databází (**Knowledge Discovery in Databases** – zkráceně **KDD**) sahá do 90. let 20. století, kdy se o této problematice začalo diskutovat na konferencích a workshopech konaných v USA. Konference byly z počátku zaměřeny spíše na oblast umělé inteligence. Získávání znalostí z databází je tedy postaveno na teoretických a praktických principech více disciplín, jakými jsou například matematika, databázové systémy, neuronové systémy, vyhledávání, vizualizace a již zmiňovaná umělá inteligence. Průmět do dílčích disciplín je znázorněn na obrázku 2.1.

Pro pojem získávání znalostí z databází existuje celá řada alternativních názvů: dolování dat (Data Mining), dolování z dat nebo z méně používaných stojí za zmínku datová archeologie. Pro úplnost je však nutné dodat, že označení dolování dat není zcela rigorózní, neboť se jedná pouze o segment procesu získávání znalostí z databází (viz. obrázek 2.3).

Podle [6] můžeme data mining definovat následovně: *„Data mining je netriviální proces zjišťování platných, neznámých, potenciálně užitečných a snadno pochopitelných závislostí v datech.“*

Z výše uvedené definice je třeba zdůraznit výrazy *netriviálnost* a *potenciální užitečnost*, protože by se nemělo jednat o proces, který lze realizovat jednoduchým SQL dotazem a zároveň by získaná informace měla být smysluplná a uplatnitelná při reálném rozhodování.



Obrázek 2.1: Průmět získávání znalostí z databází do dalších disciplín.

2.1 Business Intelligence

Přestože jsou databáze zdrojem cenných informací, málokterá společnost dovede tyto skryté informace v databázi identifikovat, extrahovat, využít ve svůj prospěch a získat tak jistý náskok před konkurencí. Nabyté informace jsou pak typicky využívány při prediktivní analýze, podpoře řízení, správy a rozhodování. Samotný proces transformování dat na informace a následné převedení těchto informací na poznatky nazýváme *Business Intelligence* (zkratka **BI**). Nejrychleji rostoucím segmentem Business Intelligence je suverénně oblast dolování z dat. V širším pojetí zahrnujeme do BI i role uživatelů, kteří s příslušnými technologiemi přicházejí do styku a klíčová rozhodování ve finální fázi záleží právě na nich.

Jako první pojem Business Intelligence definoval v roce 1989 analytik Howard Dresner:

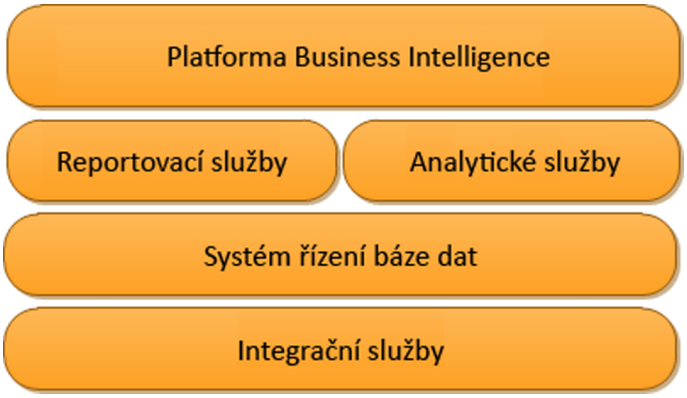
„Business Intelligence je množina konceptů a metodik, které zlepšují rozhodovací proces za použití metrik, nebo systémů založených na metrikách. Účelem procesu je konvertovat velké objemy dat na poznatky, které jsou potřebné pro koncové uživatele. Tyto poznatky potom můžeme efektivně použít například v procesu rozhodování a mohou tvořit velmi významnou konkurenční výhodu.“

Výše uvedená definice byla převzata z [6]. Hierarchická struktura Business Intelligence na platformě MS SQL Serveru 2008 je znázorněna na obrázku 2.2.

Při pohledu do současnosti nelze v oblasti Business Intelligence a dolování z dat přehlédnou nejsilnější hráče na trhu, kterými bezesporu jsou:

- SAS,
- IBM,
- Oracle,
- Microsoft.

V této bakalářské práci je pozornost věnována metodikám a principům pro dobývání dat z databází právě z pohledu společnosti Microsoft (viz. kapitola 5).



Obrázek 2.2: Hierarchická struktura Business Intelligence na platformě MS SQL Serveru 2008.

2.2 Proces získávání znalostí z dat

Proces získávání znalostí z dat je poměrně náročný a skládá se z několika vzájemně provázaných fází. Tyto fáze většinou probíhají opakovaně. Obecná podoba procesu je znázorněna na obrázku 2.3 a zpravidla obsahuje sedm kroků. První čtyři kroky lze souhrnně zařadit do fáze předzpracování. Hlavní náplní této bakalářské práce je především samotné dolování z dat a částečně i prezentace znalostí uživateli. Zde je výčet jednotlivých částí celého procesu:

1. **Čištění dat** (Data cleaning) – v této části procesu dochází k odstranění šumu a nekonzistentních dat.

2. **Integrace dat** (Data integration) – probíhá slučování a ucelení dat, která pochází z několika různých datových zdrojů. Typicky následuje uložení do datového skladu.

3. **Výběr dat** (Data selection) – jsou selektována významná data pro danou úlohu. V případě datového skladu se jedná o dimenzi, v případě relační databáze pak o tabulku.

4. **Transformace dat** (Data transformation) – nejedná se v podstatě o nic jiného, než o přizpůsobení či přeměnu dat do vhodného tvaru pro dolování (agregace, normalizace, generalizace).

5. **Dolování z dat** (Data mining) – jeden z nejdůležitějších kroků celého procesu, kdy pomocí heuristických algoritmů a metod vyhledáváme vzory a závislosti v datech.

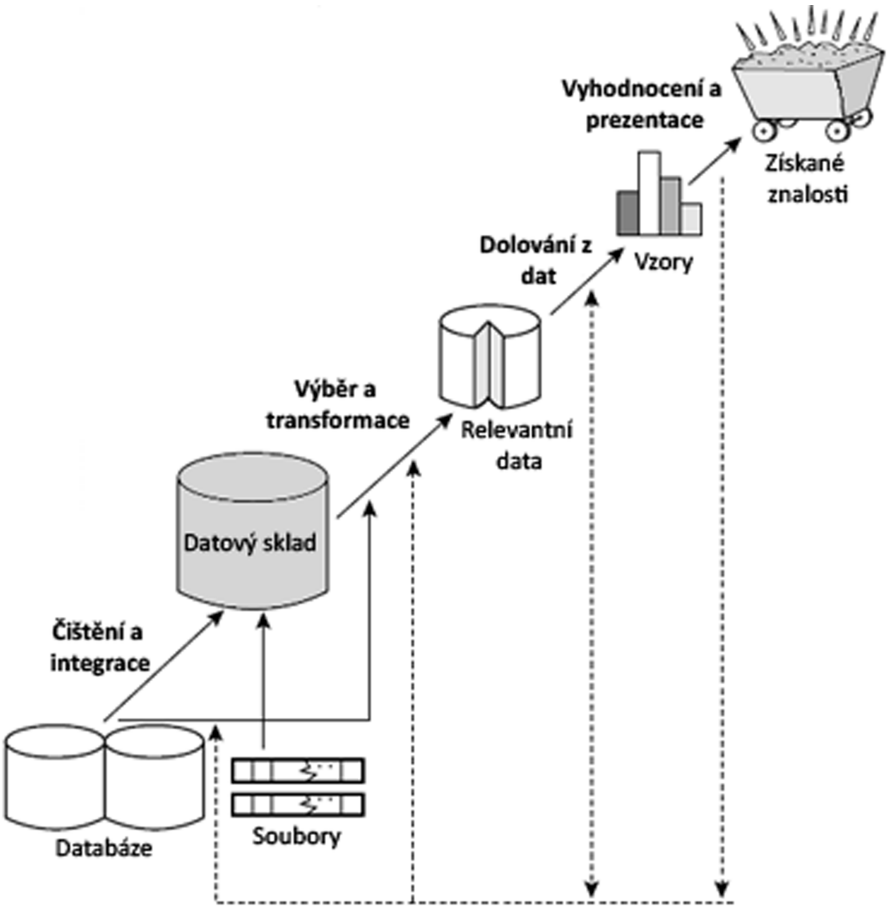
6. **Hodnocení vzorů** (Pattern evaluation) – nalezené vzory v datech se snažíme klasifikovat pomocí užitečnosti.

7. **Prezentace znalostí** (Knowledge presentation) – poslední fází je přehledná prezentace, případně vizualizace, získaných znalostí uživateli.

V praxi si však každá firma tento obecný proces modifikuje k obrazu svému. Za zmínku stojí například firma SAS, kde SAS Enterprise Miner implementuje SAS metodiku SEMMA (zkratka je složena z počátečních písmen názvů kroků celého procesu) pro projekty dolování z dat. Tato metodika odpovídá spíše technologickému pojetí a je komponována z pěti fází:

- **Sample** – volba relevantních objektů,
- **Explore** – v této části procesu dochází k redukování dat a vizuální exploraci,
- **Manipulate** – fáze zahrnující agregaci objektů a hodnot atributů i transformaci dat,
- **Model** – analyzování dat,
- **Asses** – realizování interpretace a porovnávání modelů.

Podrobnější informace o SAS metodice SEMMA lze najít na oficiálním webu firmy [10] a také částečně v literatuře [2]. Další poznatky vztahující se k procesu získávání znalostí jsou k dispozici v literatuře [2, 4, 11].



Obrázek 2.3: Proces získávání znalostí z dat (obrázek převzat z literatury [4]).

2.3 Využití dolování z dat v praxi

Data mining má v praxi velmi široké uplatnění, zejména pak v komerční podnikové sféře, do které spadá mimo jiné i marketing, kontrola kvality produktů a styk s klienty. S trochou nadsázky můžeme prohlásit, že data mining lze použít v případech, kde je reálné, alespoň po určitou dobu, uchovávat data z procesů.

Zaměříme-li se na vztah *firma – zákazník*, využíváme data mining například pro předpovídání budoucího chování zákazníka na základě jeho chování a rozhodnutí v minulosti, nebo pro zjištění, zda-li klient nehodlá přejít ke konkurenci. Podle atributů a chování, které je uloženo v databázi, firmy dokáží vytipovat správné cílové klienty i pro konkrétní marketingovou kampaň. Dalším typem využití je tzv. *analýza nákupního košíku*, jejímž cílem je nalezení zboží, které si klienti kupují současně při jednom nákupu – jedná se o tzv. asociační pravidla. Podle výsledných silných asociačních pravidel pak mohou odpovědní analytici učinit rozhodnutí týkající se například strategického rozmístění zboží v obchodě.

Čím dál častěji jsou přednosti data miningu uplatňovány i v jiných odvětvích, například v bezpečnosti při *odhalování zločinů* (nesplácení úvěru ze strany klienta) a *detekci podvodů* (neobvyklé bankovní transakce – převody extrémně vysokých částek).

V poslední době zaznamenala velmi dynamický růst i *bioinformatika*, jejíž neodmyslitelnou součástí data mining bezesporu je. Data mining aplikujeme v této oblasti mimo jiné i při analýze genetických informací, hledání skrytých souvislostí mezi jednotlivými diagnózami a při vyhledávání příznaků. Také *získávání znalostí z textů* je velmi užitečné při podrobném rozboru e-mailových zpráv a boji proti nevyžádané poště.

Na druhou stranu je však nutné podotknout, že výsledky dolování z dat nemusí být vždy přínosné a uplatnitelné v praxi. Dokonce může nastat i situace, že výsledky nebudou žádné. Proto je klíčovým faktorem úspěchu i kvalita a úplnost vstupních dat a jejich následné úpravy. Data mining musíme chápat pouze jako prostředek, či nástroj k rozhodování, ale samotné potvrzení získaných znalostí a vyvození patřičných závěrů je jednoznačně v kompetenci analytika.

Kapitola 3

Typy dolovacích úloh a algoritmů

Při volbě dolovací úlohy nebo metody je důležité si nejprve uvědomit, jaké vzory (v angličtině patterns), či modely hodláme z dat vlastně získat. Jelikož uživatelé v mnoha případech vlastně ani „netuší“, jaké vzory nebo modely pro ně mohou být zajímavé a užitečné, tak je nutné, aby aplikace pro získávání znalostí z databází byly dostatečně „pružné“, univerzální a umožňující řešení různých data miningových úloh.

Metody dolování z dat lze klasifikovat podle obecné funkcionality na:

- **Deskriptivní** – dolovací metody, které popisují obecné atributy dat (například asociační pravidla – analýza nákupního košíku).
- **Prediktivní** – dolovací metody, pomocí nichž je možné predikovat budoucí chování, a to na základě analyzování stávajících dat (například predikce poptávky klientů po konkrétním zboží nebo službách).

3.1 Popis konceptu/třídy

Jedná se o elementární typ dolovací úlohy, kde jsou data klasifikována do tříd. Tyto třídy typicky charakterizujeme rigorózním popisem, který lze vytvořit:

- **Charakterizací dat** – souhrn obecných atributů analyzované třídy. Relevantní data vyhovující dané třídě jsou většinou dostupná i prostřednictvím jednoduchého dotazu. Pro lepší pochopení si pod třídou můžeme představit produkt, jehož prodejnost na trhu klesla oproti minulému roku o 25%.
- **Diskriminací dat** – data zvolené třídy nepopisujeme obecně, ale porovnáváme hodnoty atributů s jinými třídami, v nichž se výrazně odlišují. Příklad z praxe by mohl vypadat tak, že máme porovnávat, čím se odlišují automobily, jejichž prodej v západní Evropě vzrostl o více než 5% od automobilů, jejichž prodej ve stejné lokalitě klesl o 20%.

3.2 Dolování frekventovaných vzorů, korelací a asociací

Tento typ dolovacích úloh je zaměřen především na vyhledání vzorů, vztahů a na odhalování souvislostí, které se často vyskytují v datech. Nejčastějším uplatněním tohoto typu úlohy je analýza nákupního košíku, kdy je hlavním cílem odhalit, které položky si zákazníci kupují zároveň.

Podíváme-li se na tuto úlohu z pohledu statistiky, jedná se tedy o zkoumání negativních, nebo pozitivních korelací.

„Korelace může být pozitivní a negativní. Pozitivní korelace udává, že vysoká úroveň jedné proměnné bude provázena vysokou úrovní korelační proměnné. Naopak negativní korelace udává, že vysoká úroveň jedné proměnné bude provázena nízkou úrovní korelační proměnné.“

Tato defice je převzata z [6]. Pro úplné pochopení těchto pojmů je vhodné uvést výstižné příklady. Za příklad **pozitivní korelace** můžeme považovat rostoucí poptávku po chytrých mobilních telefonech a zároveň rostoucí poptávku po mobilních aplikacích. Oproti tomu **negativní korelaci** můžeme demonstrovat na rostoucí poptávce po LCD monitorech a zároveň klesající poptávce po CRT monitorech.

Výsledkem tohoto typu úlohy mohou být tzv. asociační pravidla, která mají tvar:

$$věk(X, \text{'}> 26\text{'}) \wedge plat(X, \text{'}> 35000\text{'}) \Rightarrow koupí(X, \text{'iPad'}) \wedge koupí(X, \text{'originální pouzdro'})$$

$$[podpora = 3\%, spolehlivost = 40\%]$$

Jelikož je dolování *frekventovaných množin* a silných *asociačních pravidel* jedním z pilířů bakalářské práce, je tato problematika podrobněji popsána v části 4.1.

3.3 Klasifikace a predikce

Klasifikace a predikce spadají do kategorie prediktivních dolovacích úloh. Hlavním smyslem **klasifikace** je přiřazování dat na bázi jejich atributů do daných tříd, kterých je konečný počet. Klasifikace se skládá ze tří fází:

- **učení,**
- **testování,**
- **aplikování.**

Predikcí rozumíme předpovídání spojité hodnoty daného objektu na bázi jeho atributů (např. analytik chce predikovat celkovou sumu výdajů firmy pro budoucí rok).

Do této skupiny úloh patří i klasifikace pomocí *rozhodovacích stromů* a *Bayesovská klasifikace*, které jsou součástí implementované aplikace v praktické části bakalářské práce. Jejich obecný princip je detailněji objasněn v částech 4.2 a 4.3.

3.4 Shluková analýza

Jak již ze samotného názvu plyne, tento typ úlohy se používá pro odhalování shluků dat. Shlukování je proces, při němž jsou objekty organizovány do určitých skupin na základě podobnosti, případně odlišnosti. Objekty jsou tedy shlukovány do tříd na principu maximální podobnosti objektů v identické třídě a zároveň na principu minimální podobnosti s objekty jiných tříd. Vzniklé třídy pak tvoří shluky (klastry) – viz. obrázek 4.3.

Tento typ úlohy lze v praxi aplikovat při reklamní kampani, kde je potřeba zmapovat bydliště klientů. Do nalezených regionů (shluků) bude následně cílena reklamní kampaň společnosti.

Protože je v implementované aplikaci, v rámci praktické části, použita metoda postavená na principu shlukování, konkrétně se jedná o algoritmus *Microsoft Clustering Algorithm*, je této problematice věnována část 4.4.

3.5 Dolování odlehlých hodnot

V tomto typu dololovacích úloh je cílem nalézt v datech takové vzory, které se od ostatních výrazně odlišují. Dolování odlehlých hodnot můžeme uplatnit například v sektoru bankovnictví při odhalování tzv. „praní špinavých peněz“, které se mimo jiné vyznačuje:

- časté příkazy do zahraničních zemí (tzv. „daňové ráje“),
- časté vklady na účet a v krátkém časovém intervalu následující výběry z účtu,
- opakující se vklady na účet, které jsou však nepatrně menší než 15 000 €(tzv. „strukturování“).

3.6 Analýza evoluce

Analýza evoluce je zaměřena především na odhalování určitých pravidelností, respektive trendů, u objektů, jejichž vývoj (evoluce) se v časovém horizontu (periodicky) mění. Analýzu evoluce můžeme aplikovat v praxi třeba při sledování a vývoji akcií na akciovém trhu.

Kapitola 4

Teoretické principy vybraných dolovacích úloh a algoritmů

V rámci této kapitoly jsou blíže popsány a objasněny teoretické principy dolovacích úloh a algoritmů, které jsou implementovány v praktické části bakalářské práce. Obsah kapitoly čerpá z literatury [4, 6, 11].

4.1 Frekventované množiny a silná asociační pravidla

Pojmy *frekventovaná množina* a *asociační pravidlo* poprvé představili v roce 1993 pánové Agrawal, Imielinski a Swami. *Asociační pravidla* mohou sloužit například k odhalování jisté souvislosti mezi položkami ve vzorku dat. Na úvod si definujeme oba výše uvedené pojmy.

Frekventovanou množinou rozumíme množinu položek nebo hodnot, které se v daném vzorku dat vyskytují velmi často. Množinu lze označit za frekventovanou tehdy a jen tehdy, když má *podporu (support)* vyšší nebo rovnu minimální zadané hodnotě.

Pojem **asociační pravidlo** můžeme do jisté míry chápat jako „implikaci“ $X \Rightarrow Y$, která vyjadřuje, že pokud je v transakci obsažena položka X, tak je v transakci s určitou pravděpodobností obsažena i položka Y. Je však nutné zdůraznit, že u asociačních pravidel nemusí být pravděpodobnost implikace vždy rovna 100%.

Asociační pravidla můžeme popsat i formálnějším způsobem, a to následovně:

Nechť $\tau = \{I_1, I_2, I_3, \ldots, I_m\}$ je množina položek a nechť D je množina transakcí, kde každá transakce T je množina položek taková, že $T \subseteq \tau$. $\forall\, T\ \exists$ unikátní TID. Nechť X je množina položek. Řekneme, že transakce T obsahuje X tehdy a jen tehdy, pokud $X \subseteq T$. Asociační pravidlo je pak implikace tvaru $X \Rightarrow Y$, přičemž platí, že $X \subset T, Y \subset T$ a $X \cap Y = \emptyset$.

4.1.1 Atributy asociačních pravidel

Asociační pravidlo má dva atributy, a to *podporu (support)* a *spolehlivost (confidence)*. Tyto dva atributy reflektují užitečnost a statistický význam asociačního pravidla.

Podporou asociačního pravidla $X \Rightarrow Y$ rozumíme procentuální vyjádření počtu transakcí v D, které obsahují množinu položek $X \cup Y$. Formálně lze tedy tento atribut definovat následujícím způsobem:

$$support\big(X \Rightarrow Y\big) = P\big(X \cup Y\big) \tag{4.1}$$

Spolehlivost asociačního pravidla reprezentuje procentuální vyjádření podmíněné pravděpodobnosti, že transakce, které obsahují X zároveň obsahují také Y. Formální zápis spolehlivosti asociačního pravidla $X \Rightarrow Y$ pak vypadá s využitím pravděpodobnosti takto:

$$confidence\,(X \Rightarrow Y) = P\,(Y \mid X) = \frac{support\,(X \cup Y)}{support\,(X)} \tag{4.2}$$

Asociační pravidlo prohlašujeme za **silné** tehdy a jen tehdy, když je podpora a spolehlivost tohoto pravidla vyšší nebo rovna zadaným hodnotám (tyto hodnoty typicky stanoví uživatel).

4.1.2 Fáze dolování asociačních pravidel

Proces dolování asociačních pravidel se obecně skládá ze dvou fází:

1. **nalezení frekventovaných množin** – jedná se o množiny položek splňující podmínku minimální podpory,
2. **generování silných asociačních pravidel** – tato asociační pravidla jsou generována z frekventovaných množin a zároveň musí splňovat podmínku minimální podpory i spolehlivosti.

Ze dvou výše uvedených fází je první část výpočetně mnohokrát náročnější. Za zmínku stojí demonstrativní příklad, kde bude mít množina 100 položek.

Je třeba si uvědomit, že množina $\{I_1, I_2, I_3, \ldots, I_{100}\}$ obsahuje $\binom{100}{1} = 100$ **jednoprvkových** podmnožin: $I_1, I_2, \ldots, I_{100}$, $\binom{100}{2}$ **dvouprvkových** podmnožin:
$(I_1, I_2), (I_1, I_3), \ldots, (I_{99}, I_{100})$, a tak dále. Celkový počet obsažených kandidátů na frekventované množiny lze tedy vyjádřit vztahem:

$$\binom{100}{1} + \binom{100}{2} + \ldots + \binom{100}{100} = 2^{100} - 1 \approx 1.27 \times 10^{30}. \tag{4.3}$$

Hledání asociačních pravidel v rozsáhlých datech by bylo možné realizovat tak, že bychom postupně generovali všechny možné kombinace na levé a pravé straně pravidla. Pro každé takto vygenerované asociační pravidlo by byl proveden test, zda-li je, či není silné.

Použitelnost výše popsaného postupu je však v praxi zcela nereálná, poněvadž v průběhu výpočtu dochází k tzv. „kombinační explozi“. Hlavní nevýhoda spočívá také v exponenciální časové složitosti.

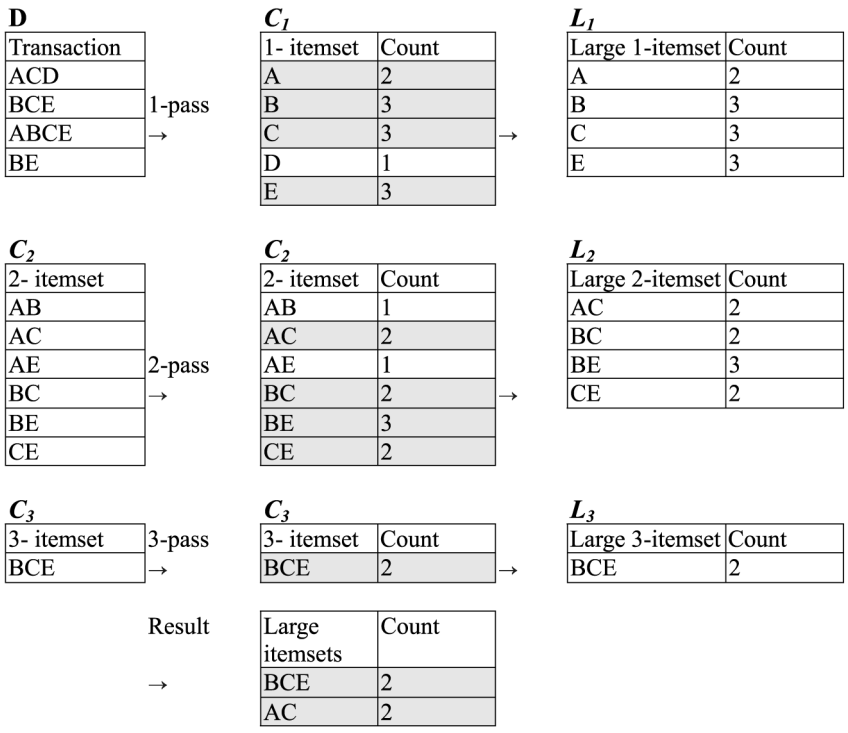
4.1.3 Algoritmus Apriori

Algoritmus Apriori slouží k získávání frekventovaných množin, z nichž jsou v následující fázi generována jednoúrovňová booleovská asociační pravidla. Jedná se o jeden z prvních algoritmů, který navrhli v roce 1994 pánové R. Agrawal a R. Srikant. Algoritmus je pojmenován podle skutečnosti, že využívá předchozích (A priori[1]) znalostí o frekventovaných množinách vygenerovaných v předešlém kroku. Jelikož v každé iteraci dochází k průchodu databází, tak je pro snížení výpočetní složitosti využívána tzv. **Apriori vlastnost**. Apriori vlastnost je splněna tehdy a jen tehdy, když je každá neprázdná podmnožina frekventované množiny taktéž frekventovaná.

Generování frekventovaných množin probíhá u algoritmu Apriori ve dvou krocích (spojovací a vylučovací), přičemž Apriori vlastnost aplikujeme ve druhém, vylučovacím, kroku.

Spojovací krok – dochází ke spojování dvou množin o stejném počtu prvků, které se liší právě v jednom prvku. Množina kandidátů C_k na frekventované množiny je generována spojením množin z L_{k-1} (operace $L_{k-1} \bowtie L_{k-1}$). Důležitým předpokladem je, aby byly položky v množině lexikograficky seřazeny.

Vylučovací krok – vygenerovaná množina kandidátů C_k je nadmnožinou množiny L_k. Je tedy nutné odstranit každou $(k-1)$-množinu, která není frekventovaná (uplatnění Apriori vlastnosti). Princip generování frekventovaných množin je zachycen na obrázku 4.1.



Obrázek 4.1: Princip funkčnosti algoritmu Apriori (obrázek převzat z [3]).

Po získání všech frekventovaných množin následuje fáze **generování silných asociačních pravidel**, které je založeno na výpočtu spolehlivosti (confidence) (viz. rovnice 4.2). Generování silných asociačních pravidel probíhá následovně:

[1] **A priori** nebo také **apriori** je latinské spojení, které znamená **předem** nebo **předchůdný**.

- Pro každou frekventovanou množinu l se zjistí všechny její neprázdné podmnožiny.
- Pro každou podmnožinu s frekventované množiny se vygeneruje pravidlo tvaru $s \Rightarrow (l - s)$.
- Podle rovnice 4.2 se vypočítá hodnota spolehlivosti pravidla.
- Pokud je spolehlivost pravidla vyšší nebo rovna zadané spolehlivosti, pak pravidlo prohlásíme za **silné**.

Níže je uvedena ukázka pseudokódu algoritmu Apriori:

```
L_1 = find_frequent_1_itemsets(D);
for(k = 2; L_{k-1} ≠ ∅; k++) {
    C_k = apriori_gen(L_{k-1});
    for each transaction t ∈ D {
        C_t = subset(C_k, t);
        for each candidate c ∈ C_t
            c.count++;
    }
    L_k = {c ∈ C_k | c.count ≥ min_support};
}
return L = ⋃_k L_k;

procedure apriori_gen(L_{k-1})
    for each itemset l_1 ∈ L_{k-1}
        for each itemset l_2 ∈ L_{k-1}
            if (l_1[1] = l_2[1]) ∧ ... ∧ (l_1[k-1] < l_2[k-1]) then {
                c = l_1 ⋈ l_2;
                if has_infrequent_subset(c, L_{k-1}) then
                    delete(c);
                else add c to C_k;
            }
    return C_k;

procedure has_infrequent_subset(c, L_{k-1})
    for each (k-1)-subset s of c
        if s ∉ L_{k-1} then
            return TRUE;
    return FALSE;
```

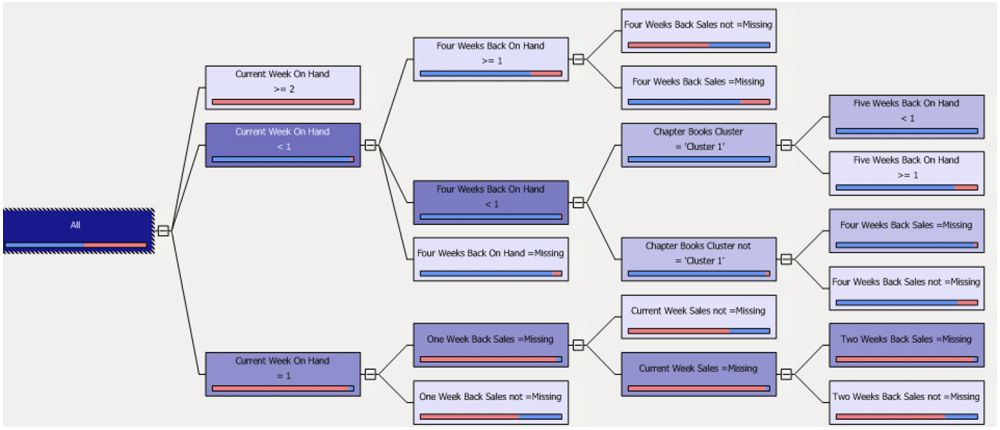
4.1.4 Výkonnostní optimalizace a varianty algoritmu Apriori

Jelikož je algoritmus Apriori **výpočetně velmi náročný**, existuje celá řada jeho vylepšených variant. Za zmínku stojí například počítání množin založené na hašování, dále pak vzorkování, redukce transakcí a další. Daleko efektivnější algoritmus pro dolování silných asociačních pravidel je algoritmus *FP-tree*. Více informací o efektivnějších variantách algoritmu Apriori a o algoritmu *FP-tree* lze získat v literatuře [4] (strana číslo 240).

4.2 Rozhodovací stromy

Algoritmus založený na principu *rozhodovacího stromu* řadíme mezi typy klasifikačních úloh (viz. část 3.3). **Rozhodovací strom** je v podstatě graf hierarchické stromové struktury (viz. obrázek 4.2). Každý jednotlivý uzel takového stromu představuje jistou vlastnost entit. Jelikož z uzlu vede **konečný počet hran**, je nutná diskretizace vlastností do konečného počtu intervalů. Vytvořený rozhodovací strom lze jednoduše převést na odpovídající klasifikační pravidla. Takové pravidlo může vypadat například následovně:

if zaměstnání = 'programátor' **and** plat = '> 30000' **then** koupíPC = **TRUE**.



Obrázek 4.2: Struktura rozhodovacího stromu v prostředí SQL Server Business Intelligence Development Studia.

Ne vždy se lze spoléhat na skutečnost, že data jsou naprosto „čistá“, a proto mohou vznikat vlivem „datového šumu“ v rozhodovacím stromu větve, které tento strom zesložiťují a snižují jeho přesnost. Pro odstranění těchto irelevantních větví se používají metody:

- **postpruning** – nejdříve je vytvořen rozhodovací strom jako celek a následně jsou větve s malým významem odstraněny,
- **prepruning** – již v průběhu vytváření rozhodovacího stromu nejsou generovány ty větve, které mají malý význam pro rozhodování.

Nelze jednoznačně prohlásit, která metoda je lepší, nebo horší. Výhodou metody postpruning je spolehlivost. Výhodou metody prepruning je zase menší výpočetní složitost. V praxi je řešením skloubení obou výše uvedených metod.

Níže je uvedena ukázka pseudokódu pro vytvoření rozhodovacího stromu (převzato z literatury [11], číslo strany 101):

function CreateTree(S, L)**: tTree;**
begin
Vytvoř nový uzel N
if(vzorky S jsou ve stejné třídě C) **return** N jako list dané třídy C
if(seznam atributů L je prázdný) **return** N jako list nejběžnější třídy v množině S
Vyber atribut A ze seznamu L s „nejvyšší rozhodovací schopností“ a odstraň jej z tohoto seznamu
Pojmenuj uzel N jménem vybraného atributu A
for(každou možnou hodnotu a_i atributu A) **do**
Vytvoř větev z uzlu N pro podmínku „$A == a_i$“
Nechť s_i je podmnožina vzorků z S, u nichž „$A == a_i$“
if(s_i je prázdná) **then** připoj k větvi list s nejběžnější třídou v množině S
else připoj k větvi podstrom vzniklý rekurz. voláním **CreateTree(**s_i**,** L**)**
end

V prostředí SQL Server Business Intelligence Development Studia je podobný typ algoritmu k dispozici pod názvem *Microsoft Decision Trees Algorithm.*

4.3 Bayesovská klasifikace

Tato metoda provádí klasifikaci, která je založena na statistice. Pro nový vzorek dat je vypočítána pravděpodobnost, s jakou patří do jednotlivých tříd. Vzorek pak zařadíme právě do té třídy, pro kterou je pravděpodobnost největší. Princip této klasifikace je založen na Bayesově větě.

Bayesův vztah pro výpočet podmíněné pravděpodobnosti, že prvek X náleží do třídy C_i vypadá následovně:

$$P(C_i \mid X) = \frac{P(X \mid C_i)P(C_i)}{P(X)} \tag{4.4}$$

Cílem je tedy nalézt maximální hodnotu výrazu $P(C_i \mid X)$. Je nutné si uvědomit, že jelikož je $P(X)$ (jmenovatel) pro vzorek X konstantní, bude celý výraz $P(C_i \mid X)$ maximální, když bude maximální právě hodnota $P(X \mid C_i)P(C_i)$ (čitatel).

Velmi výstižný příklad principu Bayesovy věty se nachází v knize [6] (číslo strany 280):

„Dobrým hypotetickým příkladem pro vysvětlení Bayesovy věty je novorozenec, který pozoruje, zda bude v noci venku zima. První den to neumí posoudit, protože to ještě nikdy nezažil, a tedy pravděpodobnost bude 0,5, tedy 50%. Každý další den když nastane noc, se jeho odhad pravděpodobnosti tohoto jevu zpřesňuje, v tomto případě zvyšuje.“

Mezi výhody tohoto algoritmu patří bezesporu snadná implementace, rychlost a v řadě případů také dosažení velmi dobrých výsledků. Poměrně citelnou nevýhodou však zůstává předpoklad, že dané atributy jsou na sobě nezávislé (což je v praxi málokdy splnitelné). Algoritmus podporuje predikci pouze diskrétních atributů.

Originální název algoritmu zakomponovaného v praktické části je *Microsoft Naive Bayes Algorithm.*

4.4 Shlukování

Jak již bylo zmíněno v části 3.4, shlukování je proces rozdělování objektů do tříd na základě jejich podobnosti. Třídy jsou pak složeny z takových objektů, které jsou si v rámci identické třídy co nejvíce podobné a zároveň jsou co nejvíce odlišné od objektů jiných tříd. Pokud budeme posuzovat shlukování z hlediska strojového učení, pak se jedná o učení bez učitele.

Dále se budeme věnovat principu metody *Expectation – Maximization* (EM) a metody *K-means*, protože algoritmus *Microsoft Clustering*, který je používán v praktické části, je založen právě na těchto dvou metodách.

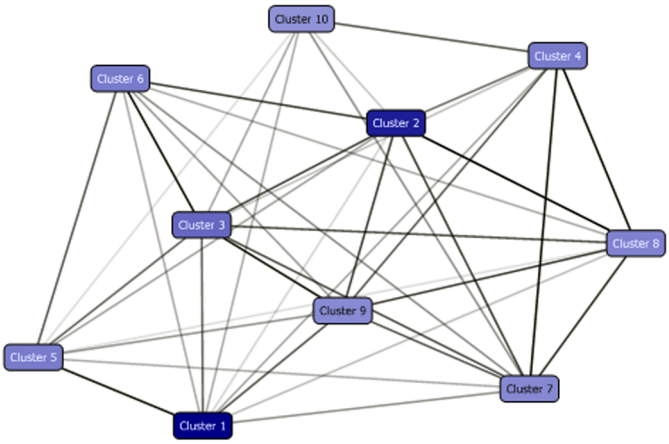
Metoda **Expectation – Maximization (EM)** patří do skupiny metod založených na modelech. Shluky jsou zde reprezentovány pomocí parametrizovaných pravděpodobnostních distribučních funkcí a datovou množinou je pak směsice takovýchto distribučních funkcí. Samotný model je pak komponován z k pravděpodobnostních distribučních funkcí, přičemž každá funkce představuje právě jeden klastr. Jádro této metody je složeno ze dvou cyklicky se opakujících kroků:

- **expectation step** – dochází k určení pravděpodobnosti, že daný objekt přísluší ke klastru,
- **maximization step** – využití pravděpodobností z předchozích kroků pro výpočet nových parametrů.

Velkou výhodou této metody je její jednoduchost, rychlost a snadná implementace.

Metoda **K-means** patří do skupiny metod založených na rozdělování. Funguje na podobném principu jako metoda Expectation – Maximization. Každá třída je u tohoto algoritmu reprezentována prostřednictvím **fiktivního centrálního bodu**. Nejprve je střed zvolen náhodně, ale při dalších krocích je již vypočítán na základě vzdáleností mezi objekty, které do třídy patří. Pro úplnost je důležité dodat, že každý objekt může patřit pouze do jedné třídy. Algoritmus je ukončen, pokud již nenastane přesun žádného objektu.

Jednou z nevýhod metody je, že neumožňuje nalézt shluky různé velikosti a nekonvexního tvaru. Důležitost je kladena i na kvalitu dat, protože výskyt odlehlých hodnot a šumu značně zkreslí finální rozložení klastrů.



Obrázek 4.3: Ukázka shluků v prostředí SQL Server Business Intelligence Development Studia.

Kapitola 5

Databázový MS SQL Server 2008

V předešlých kapitolách byly objasněny teoretické principy dolování z dat a nyní se naše pozornost bude ubírat směrem ke konkrétní platformě, na níž je realizována databázová část aplikace Data Miner 2011. Jedná se o systém MS SQL Server 2008. MS SQL Server 2008 je složen z více komponent, které si postupně představíme:

- Database Engine,
- Business Intelligence,
  - Integration Services,
  - Reporting Services,
  - Analysis Services,
    - OLAP,
    - Data Mining.

Jako podklad pro tvorbu této kapitoly sloužila literatura [5, 6, 8], ve které se čtenář může dozvědět o MS SQL Serveru 2008 detailnější informace.

5.1 Databázový modul

Database Engine je jedna z klíčových služeb SQL Serveru, která umožňuje manipulaci s daty (načítání, zpracování, ukládání, zabezpečení, ...). Slouží také k vytváření velmi výkonných databázových aplikací pro OLTP (online transaction processing), nebo OLAP (online analytic processing). Prostřednictvím Database Engine lze mimo jiné například vytvářet indexy, pohledy, uložené procedury a triggery.

5.2 Modul Business Intelligence

Druhým segmentem je **Business Intelligence**, což je z pohledu MS SQL Serveru v podstatě „zastřešení“ nad integračními, analytickými a reportovacími službami. O BI v obecném slova smyslu pojednává část 2.1.

5.2.1 Integrační služby

Modul **Integration Services** slouží především ke sběru dat z heterogenních systémů a také k zavádění dat z databází do datových skladů. Jedná se o balík služeb, v němž jsou zahrnuty různorodé úkoly, prostřednictvím kterých je možné například stahovat soubory z FTP serverů, importovat/exportovat soubory do/z databáze a v neposlední řadě lze pomocí této služby interagovat s jinými webovými službami. Důležitou zabudovanou funkcí je i schopnost kopírovat objekty SQL Serveru. Jestliže se v integračních službách nenachází taková úloha, kterou uživatel požaduje, tak si může s využitím nástrojů VSTA (Visual Studio Tools for Applications) takovou úlohu vytvořit a přizpůsobit na míru. Nejdůležitější novinkou u integračních služeb ve verzi MS SQL Server 2008 je především nový přístup při správě vláken datových toků. U víceprocesorových a vícejádrových systémů tím dochází k rapidnímu zvýšení výkonu.

5.2.2 Reportovací služby

Cílem **Reporting Services** je generování reportů a sestav z databází. Tyto reporty a sestavy pak slouží jako podklady pro podporu rozhodování zaměstnanců. Propracované schéma architektury Reporting Services na platformě MS SQL Server 2008 a srovnání oproti verzi 2005 je k dispozici v knize [6] (číslo strany 327).

5.2.3 Analytické služby

Služba **Analysis Services** se dá logicky rozdělit na dvě složky, a to na **OLAP** a **Data Mining**. Prostřednictvím služby OLAP můžeme vytvářet a spravovat multidimenzionální kostky a také se na tyto kostky dotazovat (MDX). Klíčovou částí analytických služeb pro realizaci aplikace Data Miner 2011 však byla část Data Mining. V tomto modulu jsou zabudovány všechny významné dolovací úlohy a algoritmy, pomocí kterých lze realizovat proces dolování z dat:

- Rozhodovací stromy (Microsoft Decision Trees)
- Shlukování (Microsoft Clustering)
- Sekvenční shlukování (Microsoft Sequence Clustering)
- Asociační pravidla (Microsoft Association)
- Časové řady (Microsoft Time Series)
- Neuronové sítě (Microsoft Neural Network)
- Naive Bayes (Microsoft Naive Bayes)
- Lineární regrese (Microsoft Linear Regression)
- Logistická regrese (Microsoft Logistic Regression)

Tvorbu dolovacího modelu, který je založen na některém z výše uvedených algoritmů, je možné uskutečnit několika různými způsoby, jako například použitím DMX jazyka nebo využitím průvodce v prostředí BI Development Studia (více viz. část 7.1.2).

Kapitola 6

Specifikace požadavků a návrh aplikace

Hlavním cílem této bakalářské práce je navrhnout a implemetovat aplikaci, která bude provádět získávání znalostí z databází s využitím metod zabudovaných v rámci MS SQL Serveru 2008 a zvolené metody, která na této platformě podporována není a bude tedy implementována plně ve vlastní režii. Nejprve jsou v této kapitole specifikovány požadavky na výsledný systém, vybrány technologie pro realizaci projektu a následně je představen návrh klientské a serverové části aplikace.

6.1 Neformální specifikace požadavků na systém

Výsledným produktem by měla být desktopová aplikace, která uživateli zprostředkuje získávání znalostí z databází. Jelikož jsou výstupem vydolované znalosti, které bude třeba přehledně a jasně prezentovat, nejlépe pomocí grafů či tabulek, je jisté, že aplikace musí disponovat grafickým uživatelským rozhraním (GUI). Dále je pak nutné uvažovat, že si uživatel bude chtít vydolované znalosti exportovat a trvale uložit. Pro tento případ je vhodná funkcionalita exportu získaných znalostí do jednoduchého souborového formátu (například CSV). Uživatel by také měl mít možnost upřesnit, jaké atributy vydolovaných znalostí chce zobrazit (možnost pokládat vlastní dotazy). Poněvadž může nastat i situace, že proces dolování znalostí bude trvat neúměrně dlouho (aktuální vytížení serveru, náročnost výpočtu, ...), musí být tato operace skrze GUI kdykoli v jejím průběhu bezpečně přerušitelná a opětovně spustitelná. Zároveň je žádoucí bezproblémové používání grafického uživatelského rozhraní při právě probíhajícím procesu dolování (schopnost SW vykonat více operací souběžně). V projektu bude nepochybně zakomponována i nápověda a kontakt pro případ potíží s používáním SW.

Shrňme si tedy všechny klíčové požadavky, které by výsledný produkt měl splňovat:

- desktopová aplikace pro získávání znalostí z databází ⇒ přehledné, jednoduché a uživatelsky přívětivé GUI,
- přehledná prezentace/vizualizace vydolovaných znalostí (tabulky, grafy),
- export a uložení vydolovaných znalostí (CSV),
- možnost vytvářet vlastní dotazy pro dolování,

- bezpečné přerušení procesu dolování a jeho opětovné spuštění,
- zobrazení nápovědy/manuálu a kontaktu pro uživatele.

6.2 Volba technologií pro realizaci aplikace

V zadání bakalářské práce je jako databázový server určen MS SQL Server. Jasnou volbou pro implementaci aplikace je jazyk C# na platformě .NET Framework verze 4.0 a to hned z několika důvodů:

- jazyk C# vyvinula stejná firma jako SQL Server 2008 (kompatibilita, podpora, kvalitní dokumentace),
- .NET Framework – typová bezpečnost, podpora tříd, vlastností, metod, konstruktorů, polymorfismu,
- jazyk C# je integrován do vývojového prostředí Visual Studio .NET (v prostředí MS VS je mimo jiné i velmi rychlý a pohodlný vývoj GUI),
- jazyk je čistě objektově orientovaný,
- automatická správa paměti (garbage collector) a další.

Výsledná desktopová aplikace v jazyce C# bude typu WinForms (disponuje grafickým uživatelským rozhraním). Pro dotazování se na modely ze strany klienta bude použita technologie ADOMD.NET zahrnující i knihovnu Microsoft.AnalysisServices.AdomdClient.dll. Prostřednictvím této knihovny lze tedy zasílat na model dotazy zapsané například v jazyce DMX (Data Mining Extension).

6.3 Návrh serverové části

Ze zadání a specifikace vyplývá, že mají být použity dolovací metody zabudované v MS SQL Serveru 2008. Je tedy žádoucí, aby byly na straně serveru vytvořeny data miningové modely, se kterými bude klient „komunikovat“ prostřednictvím DMX (Data Mining Extension) dotazů. Tvorba takového modelu je popsána v části 7.1.2. Dále pak bude k dispozici transakční databáze, aby mohla klientská část aplikace provádět nad touto databází tzv. analýzu nákupního košíku s využitím algoritmu Apriori (viz. část 4.1.3).

6.3.1 Zdrojová data

Jako zdrojová data pro celý projekt poslouží jednak **databáze Adventure Works DW 2008** a **databáze** pořízená z **internetového obchodu společnosti NSU.cz**, která se zabývá prodejem dílů na historické motocykly značky NSU.

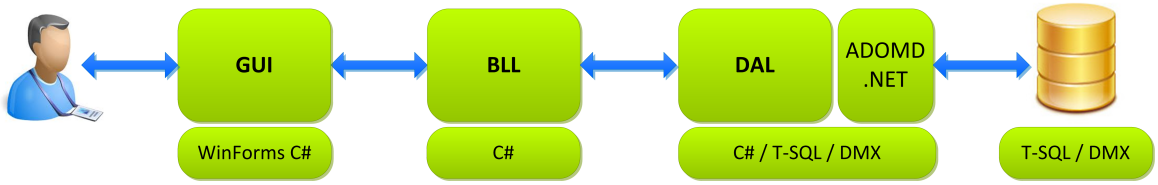
V případě Adventure Works DW 2008 se jedná o databázi, jejíž data jsou založena na fiktivní národní společnosti, která vyrábí a prodává jízdní kola po celém světě. Společnost Microsoft poskytuje tuto ukázkovou databázi volně ke stažení. Právě nad Adventure Works DW 2008 budou dolovací modely vytvořeny, trénovány a testovány, aby pak následně bylo možné prostřednictvím těchto modelů predikovat potenciální kupce produktů.

Databáze internetového obchodu NSU.cz vychází narozdíl od Adventure Works DW 2008 z reálných uskutečněných transakcí (nejedná se tedy o nějaká náhodně vygenerovaná

data), a proto je tato transakční databáze vhodná pro metodu Apriori, která bude implementována ve vlastní režii. Výsledkem pak budou silná asociační pravidla vyjadřující, jaké náhradní díly na motocykly si klienti kupovali nejčastěji souběžně.

6.4 Návrh klientské části

Při realizaci klientské části se přímo nabízí použití vícevrstvé architektury. Jednak proto, že aplikace bude mít grafické rozhraní, ale také bude přistupovat k datům uloženým v databázi na straně serveru. V následující části si architekturu aplikace Data Miner 2011 představíme.



Obrázek 6.1: Model aplikace Data Miner 2011.

6.4.1 Třívrstvá architektura aplikace

Ideální volbou pro aplikaci Data Miner 2011 je architektura skládající se ze tří vrstev. Každá z vrstev zapouzdřuje svoji implementaci a navenek poskytuje pouze jisté rozhraní. Tento přístup má několik podstatných výhod:

- znovupoužitelnost libovolné vrstvy v jiném projektu,
- možnost nahrazení kterékoli vrstvy vrstvou jinou (při dodržení stanoveného rozhraní),
- přehledné členění celého projektu na více nezávislých modulů, které mohou být implementovány odděleně.

Pokud bychom posuzovali projekt jako celek (klientská a serverová část dohromady), mohli bychom prohlásit, že ve výsledku máme vrstvy čtyři, a to: prezentační, aplikační, datovou a serverovou. Do serverové části by spadaly vytvořené data miningové modely. V tomto úseku se však z pohledu projektu zaměřujeme pouze na klientskou část. Nyní si představíme funkce jednotlivých vrstev, které jsou znázorněny na obrázku finálního modelu aplikace (viz. obrázek 6.1).

Prezentační vrstva (Presentation Layer - GUI) má za cíl zprostředkovávat interakci mezi uživatelem a aplikací. Reaguje na zadané vstupy, zobrazuje a prezentuje výsledná data. V prezentační vrstvě jsou zahrnuty i funkce uživatelského rozhraní.

Úkolem **aplikační vrstvy** (Business Logic Layer - BLL) je provádění výpočtů a zpracování dat, včetně validace uživatelských vstupů. Tato vrstva obsahuje tzv. business logiku, nebo také jádro celé aplikace. Algoritmus Apriori bude implementován právě v této vrstvě. Aplikační modul slouží také jako prostředník mezi prezentační a datovou vrstvou.

Datová vrstva (Data Access Layer - DAL) slouží pro přístup k datům, která jsou uložena typicky na straně serveru v databázi, ale neprovádí s daty žádné výpočty (to je úkolem aplikační vrstvy). Na zpřístupněná data pak předává referenci aplikační vrstvě.

Kapitola 7

Implementace aplikace Data Miner 2011

Tato kapitola si klade za cíl přiblížit čtenáři průběh implementace celého projektu postupně od serverové ke klientské části. Jelikož se jedná o poměrně složitý a rozsáhlý projekt, je popis implementace značně abstrahován. Není tedy prioritou zacházet do nejmenších detailů, ale naopak zdůraznit klíčové kroky implementace.

7.1 Implementace serverové části

V serverové části je nejprve představena struktura transakční databáze, nad kterou operuje algoritmus Apriori a také je zde popsána tvorba dolovacích modelů.

7.1.1 Struktura a import databáze Apriori

Databáze Apriori je založena na reálných uskutečněných transakcích, které byly poskytnuty z internetového obchodu společnosti NSU.cz. Jedná se o typ transakční databáze skládající se ze dvou tabulek. V tabulce ***dbo.TransactionsNSU*** je v prvním sloupci uložena číselná hodnota unikátního identifikátoru transakce (*TransactionID*) a ve druhém sloupci je celočíselný identifikátor produktu (*ProductID*). Druhá tabulka, ***dbo.ProductsNSU***, pak slouží k získání konkrétního názvu výrobku (*ProductName*) prostřednictvím unikátního identifikátoru (*ProductID*). Menší problém nastal při importu databáze na MS SQL Server 2008. Jelikož byla databáze dodána v textovém souboru, nejprve ji bylo třeba převést do formátu CSV. Poté již stačilo vytvořit jednoduchý T-SQL skript, který s využitím příkazu *BULK INSERT ...* importoval soubor s takto upravenými daty do vytvořených tabulek. Pro případ, že by v budoucnu bylo potřeba databází opětovně na MS SQL Serveru vytvořit, je uložen na přiloženém DVD skript *AprioriScript.sql*, který po spuštění vytvoří celou databázi včetně tabulek a zároveň je i naplní daty. Z této databáze jsou pak pomocí algoritmu Apriori dolována silná asociační pravidla s odpovídající podporou a spolehlivostí.

7.1.2 Proces tvorby data miningových modelů

Nejdůležitější částí na straně serveru jsou jednoznačně data miningové modely, které jsou nasazeny konkrétně na MS Analysis Serveru. Modely pracují nad databází fiktivní společnosti Adventure Works, zabývající se prodejem jízdních kol. Nad touto databází se tedy přímo nabízí řešit typickou otázku predikce, zda-li si zákazník produkt koupí, či nikoli.

Celkem je vytvořeno pět data miningových modelů, které využívají zabudované algoritmy: *Microsoft Decision Trees*, *Microsoft Naive Bayes* a *Microsoft Clustering*. V aplikaci jsou však zpřístupněny pouze obecnější tři modely. Zbylé dva rozlišují pohlaví klientů a do budoucna by nebyl žádný problém tyto modely do aplikace také zakomponovat.

Modely lze na straně serveru vytvořit několika způsoby. Prvním způsobem je napsat skript v jazyce DMX a tento skript následně spustit v prostředí Analysis Serveru. Ve skriptu musí být zahrnuty všechny potřebné údaje, jako definice zdrojových dat, datových pohledů a vytvoření konkrétního modelu. Pseudokód pro vytvoření konkrétního modelu, který bude založen na algoritmu *Microsoft Association Rules* by vypadal v jazyce DMX následovně:

```
CREATE MINING MODEL MyNewAssociationModel (
      OrderNumber TEXT KEY,
      [Products] TABLE PREDICT (
            [Model] TEXT KEY
      )
)
USING Microsoft_Association_Rules (MIN_PROBABILITY = 0.1, MIN_SUPPORT = 0.01).
```

Dalším způsobem, jak je možné modely na straně serveru vytvořit je použití značkovací jazyk XML. Princip je stejný, jako v předchozím případě, jen zapisujete potřebné údaje logicky pomocí jazyka XML a poté odesíláte na server. Třetím způsobem je využití Business Intelligence Development Studia. Právě poslední jmenovaný způsob je aplikován v této bakalářské práci. Celý proces od založení projektu, definování datových zdrojů, až po nasazení modelů v BI Development Studiu je popsán v následujících částech.

Nejdříve je potřeba v prostředí BI Development Studia vytvořit nový projekt, který bude typu *Analysis Services Project*. Bezprostředně po založení projektu je vhodné nastavit cílový server (v případě lokálního použití logicky *localhost*).

Definování datových zdrojů

Abychom vůbec mohli dolovací modely začít vytvářet, tak musíme definovat zdrojová data (výběr databáze). Pro tento účel je v rámci BI Development studia zabudován průvodce, který vás procesem volby zdrojových dat bezpečně provede. Při tomto procesu bude po uživateli vyžadována volba poskytovatele dat, název serveru, na němž jsou data uložena, jméno zdrojové databáze a především volba přihlašování na server (pro lokální účely je vhodné ponechat volbu *Use Windows Authentication*). Jakmile je tento proces úspěšně dokončen, tak se v pravém panelu ve složce *Data Sources* zobrazí nový právě definovaný datový zdroj (viz. obrázek 7.1).

Definování datových pohledů

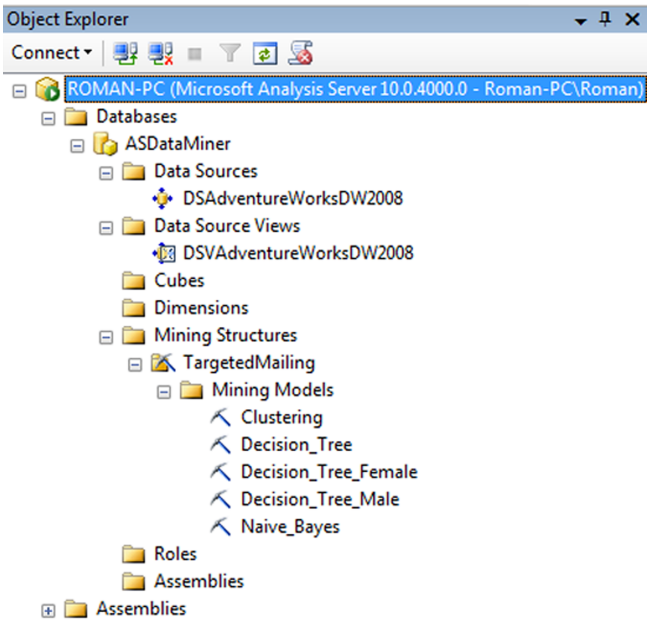
Pokud je datový zdroj z předchozího kroku úspěšně vytvořen, můžeme přejít k tvorbě datových pohledů (pozor, bez datového zdroje datový pohled nelze vytvořit). Tak jako v předcházejícím kroku si spustíme průvodce a ten nám nabídne volbu konkrétních tabulek a pohledů z databáze. Stačí tedy zvolit odpovídající hodnoty a vytvořený pohled si pojmenovat. Výsledek se po dokončení opět zobrazí v pravém panelu ve složce *Data Source Views*.

Vytvoření konkrétního data miningového modelu

Po nadefinování datových zdrojů a pohledů přichází na řadu hlavní část – tvorba dolovacího modelu. Opět si spustíme průvodce a první věcí, kterou musíme učinit je volba dolovacího algoritmu, na jehož principu bude model fungovat. V nabídce máme na výběr celkem z devíti zabudovaných algoritmů (v našem případě zvolíme *MS Decision Trees*). Po volbě algoritmu nastává důležitá fáze, a to selekce klíčového sloupce, pomocí jehož hodnoty lze entitu jednoznačně odlišit od ostatních (ideální je primární klíč). Dále zbývá definovat vstupní atributy a konečně také atribut, jehož predikci od modelu požadujeme. Je-li třeba, tak ještě manuálně upravíme datové typy sloupců. Model prochází i fázemi učení a testování. Průvodce vás tedy vyzve pro upřesnění a procentuální vyjádření, kolik dat bude sloužit pro učení a kolik pro testování. Vybereme vhodné pojmenování modelu a průvodce ukončíme.

To ale není vše. Doposud jsem pouze vytvořili analytický projekt (*solution*) v prostředí BI Development Studia. Nesmíme však opomenout celý projekt, včetně modelů, sestavit a nasadit na Analysis Server. Nejprve tedy zvolíme možnost *Build* a následně možnost *Deploy Solution*. Pokud oba kroky proběhly bez chyb, můžeme se připojit na Analysis Server a ověřit si, že je zde celý projekt včetně datových zdrojů, pohledů a vytvořených modelů úspěšně nasazen (viz. obrázek 7.1).

Zajímavé je i vzájemné porovnání všech vytvořených modelů z výkonnostního hlediska. Přece jen modely vykonávají stejnou činnost, ale každý je postaven na principech jiného dolovacího algoritmu. Z grafu, který je znázorněn na obrázku 7.2 vyplývá, že nejvhodnějším algoritmem pro řešení dané dolovací problematiky je jednoznačně algoritmus MS Decision Trees (je nejblíže ideálnímu modelu).



Obrázek 7.1: Struktura projektu ASDataMiner nasazeného na Analysis Serveru.

7.2 Implementace klientské části

Implementaci klientské části si nejprve představíme po jednotlivých vrstvách, od datové vrstvy, přes použití knihovny *AdomdClient.dll*, až po vrstvu prezentační. Následně je popsána implementace algoritmu Apriori, který je realizován především v aplikační vrstvě, ale i datová vrstva poskytuje třídu umožňující algoritmu přístup k transakční databázi. Ke konci této části je popsána i práce s vlákny s využitím třídy *BackgroundWorker*. Při implementaci desktopové aplikace Data Miner 2011 v jazyce C# posloužila jako studijní materiál literatura [1, 7]. V popisu implementace klientské části jsou pro přehlednost uváděny pouze názvy metod bez jejich typů a parametrů. Přesné definice metod včetně typů, parametrů a plných názvů jsou k dispozici pro každou vrstvu postupně ve vygenerovaných dokumentacích z prostředí Visual Studia 2010. Konkrétně se jedná o následující soubory na přiloženém DVD: *DAL.xml, BLL.xml, GUI.xml*.

7.2.1 Vývojové prostředí a založení projektu

Jelikož je Data Miner 2011 desktopovou aplikací typu WinForms v jazyce C#, která je postavena nad .NET Frameworkem verze 4.0 v prostředí MS Windows, tak byla volba integrovaného vývojového prostředí (IDE) velmi jednoduchá. Projekt byl tedy vyvíjen v prostředí MS Visual Studio 2010 Professional. Další výhodou tohoto vývojového prostředí je fakt, že je pro studenty i vyučující v plné verzi volně ke stažení, a to od společnosti Microsoft DreamSpark. Ve Visual Studiu 2010 je také poměrně propracovaná podpora pro tvorbu GUI a mnoho dalších funkcí, které vývoj značně usnadňují.

Na začátku samotného vývoje je nutné založit a pojmenovat nové *solution*. V novém *solution* jsou pak založeny tři projekty (vrstvy). Konkrétně *GUI*, *BLL* a *DAL*. Aby byly projekty mezi sebou schopny „komunikovat“, je potřeba přidat každé vrstvě referenci na vrstvu nižší. Nejnižší vrstvě, čili *DAL*, přidáme ještě referenci na knihovnu *AdomdClient.dll*, jenž je nezbytná pro komunikaci s Analysis Serverem. Celá hierarchie referencí mezi projekty v *solution* bude vypadat následovně: *GUI* → *BLL* → *DAL* → *AdomdClient.dll*).

7.2.2 Datová vrstva – Data Access Layer

Jak již bylo řečeno v předchozí kapitole, úkolem datové vrstvy je poskytovat přístup k datům uloženým typicky v databázi (mohou však být uložena například v souboru typu XML nebo také v textovém souboru). Projekt Data Miner 2011 má datovou vrstvu složenou ze dvou hlavních tříd: *AnalysisDAL.cs* a *AprioriDAL.cs*.

První třída, ***AnalysisDAL.cs***, zpřístupňuje data, která jsou vydolována na straně Analysis Serveru data miningovými modely. Konkrétně v sobě třída obsahuje klíčovou metodu *LoadAnalysisData()*, která přijímá od aplikační vrstvy jako parametr dotaz typu DMX. Tento dotaz odešle s využitím knihovny *AdomdClient.dll* na Analysis Server, přijatá vydolovaná data uloží do tabulky a na tabulku předá referenci aplikační vrstvě. Aplikační vrstva si pak ve vlastní režii provede nad tabulkou příslušné operace.

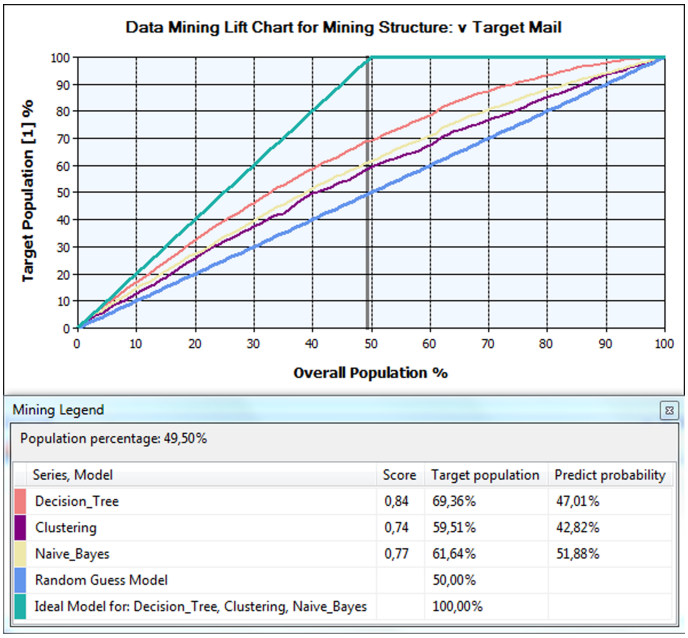
Třída ***AprioriDAL.cs***, která poskytuje přístup k datům pro algoritmus Apriori je již podstatně složitější. Obsahuje v sobě několik velmi důležitých metod. Nebudeme si však uvádět výčet všech metod, ale zmíníme se jen o těch klíčových.

V úplném začátku algoritmu Apriori je volána metoda z datové vrstvy, konkrétně se jedná o metodu *ScanForFrequent1Itemset()*, která zjistí pomocí vygenerovaného T-SQL dotazu jednoprvkové frekventované množiny. Tyto množiny ukládá do struktury *Dictionary<List<int>, double>* a vrací jako výsledek.

Další velmi významnou metodou této třídy je metoda *GetCandidateSupport()*, která pracuje tak, že jako parametr přijme množinu kandidátů a prostřednictvím dynamicky vygenerovaného T-SQL dotazu vrátí počet transakcí, v nichž se vyskytují zároveň všichni kandidáti. Pro jasnější pochopení principu metody si ukážeme krátký příklad. Nechť je metodě předána následující vstupní množina kandidátů: $\{8, 13, 37\}$. Pak metoda dynamicky vygeneruje a odešle na server následující T-SQL dotaz:

```
SELECT COUNT (*)
FROM [dbo].[TransactionsNSU] [0]
      LEFT JOIN [dbo].[TransactionsNSU] [1]
            ON [0].TransactionID = [1].TransactionID
      LEFT JOIN [dbo].[TransactionsNSU] [2]
            ON [1].TransactionID = [2].TransactionID
WHERE [0].ProductID = 8 AND [1].ProductID = 13 AND [2].ProductID = 37.
```

Více metod si již rozebírat nebudeme, ale slouží například k získání názvu produktu na základě jeho identifikátoru, nebo ke zjištění celkového počtu unikátních transakcí v databázi, a další.



Obrázek 7.2: Porovnání přesnosti vytvořených modelů.

Použití knihovny AdomdClient.dll

Jak již bylo v předcházejícím textu uvedeno, datová vrstva využívá služeb knihovny *AdomdClient.dll*. Pojďme si nyní představit třídy z této knihovny, jejichž implementace byla pro práci na projektu potřebná. Abychom mohli začít dolovat z Analysis Serveru znalosti, musíme s tímto serverem nejprve navázat spojení, a to tak, že vytvoříme novou instanci třídy *AdomdConnection* a vyvoláme odpovídající metodu *Open()*. Jakmile máme spojení úspěšně

navázáno, můžeme zaslat na server požadavek (v našem případě DMX dotaz). Vytvoříme tedy instanci třídy *AdomdCommand* se dvěma parametry: text DMX dotazu[1] a objekt vytvořeného spojení. Odeslání našeho dotazu zajistíme vyvoláním metody *ExecuteReader()* a očekávaný výsledek uložíme do proměnné *AdomdDataReader reader*. Pro ukázku a lepší pochopení je vhodné uvést krátký úsek kódu:

```
AdomdConnection conn = new AdomdConnection(...);
conn.Open();
AdomdCommand cmd = new AdomdCommand(commandText, conn);
AdomdDataReader reader = cmd.ExecuteReader(CommandBehavior.CloseConnection);
```

Ucelený přehled všech možností využití knihovny včetně ukázkových příkladů naleznete na webových stránkách [9].

7.2.3 Aplikační vrstva – Business Logic Layer

Stejně jako datová, i aplikační vrstva má dvě hlavní třídy: ***AnalysisBLL.cs*** a ***AprioriBLL.cs***. Třída *AnalysisBLL.cs* je vytvořena pro obsluhu analytických procesů. V této třídě je pro každý data miningový model implementována metoda, uvnitř které je sestaven konkrétní DMX dotaz. Tento DMX dotaz je předáván datové vrstvě a jako výsledek je vrácena tabulka naplněná daty. Aplikační vrstva s tabulkou nedělá nic jiného, než že na ni předá referenci prezentační vrstvě, která ji zobrazí uživateli. Jelikož byla v kapitole 6 požadována i funkcionalita na tvorbu vlastního DMX dotazu ze strany uživatele prostřednictvím GUI, je v této třídě zakomponována i metoda *ExecuteUserCommand()*. Tato metoda přijímá z grafického rozhraní zdrojový DMX kód vytvořený uživatelem a odešle jej na server. Výsledek pak vrací stejným způsobem, jako v případě bežného předdefinovaného dotazu. Je samozřejmé, že pokud uživatel zadá syntakticky nebo sémanticky chybný DMX dotaz, což bude celkem častá situace, tak se odesílá do prezentační vrstvy uživateli zpráva s podrobným popisem vzniklé chyby.

Implementace algoritmu Apriori

Třída *AprioriBLL.cs* obsahuje jádro celého algoritmu Apriori. Algoritmus je nastartován vyvoláním metody *Apriori(double minSupport, double minConfidence)*, přičemž parametry minimální podpora a spolehlivost zasílá prezentační vrstva, respektive zadává uživatel. Jak již bylo uvedeno v části 4.1.3, algoritmus probíhá ve dvou fázích.

V první fázi jsou vygenerovány všechny frekventované k-množiny. Generování všech k-množin má v režii metoda *RunFrequentItemsets()*. Nejprve si s využitím dalších metod a datové vrstvy vygeneruje jednoprvkové, až k-prvkové kandidátní množiny, z nichž jsou postupně generovány frekventované množiny. V mezikrocích jsou vždy vyřazovány ty podmnožiny, které nesplňují Apriori vlastnost nebo nevyhovují stanovené minimální podpoře. O testování Apriori vlastnosti se stará metoda *TestAprioriProperty()*. Generování k-množin je ukončeno v případě, že je kandidátní množina prázdná ⇒ není dále z čeho generovat.

Jakmile jsou všechny frekventované k-množiny vydolovány, tak nastává výpočetně méně náročnější fáze – tvorba silných asociačních pravidel. Tento krok má v popisu práce me-

[1] DMX dotaz si můžeme buď napsat vlastní, nebo jej lze namodelovat v prostředí BI Development Studia.

toda *BuildAssociationRules()*, která postupně pro každou neprázdnou podmnožinu frekventované množiny sestaví asociační pravidlo. Hledání všech neprázdných podmnožin dané množiny implementuje metoda *GetAllSubsets()*. Podle vzorce 4.2 je pak vypočtena spolehlivost sestaveného pravidla. V případě, že je spolehlivost dostatečně vysoká, řadíme pravidlo mezi silná asociační pravidla.

Jelikož jsou v pravidlech uvedeny jen kódy jednotlivých produktů, které uživateli ve výsledku nic přínosného nesdělují, je odeslán požadavek na datovou vrstvu, aby zjistila z databáze názvy všech odpovídajících náhradních dílů. V pravidlech jsou pak všechny kódy nahrazeny plnými názvy tak, aby uživatel lehce rozpoznal o jaký náhradní díl se jedná. Čili výsledné pravidlo má například tvar:

gumove_operky_na_nadrz $\Rightarrow$ *drzak_gumove_nakolenice_na_nadrz [sup: 4.05%, con: 60%].*

Celý popis implementace algoritmu je samozřejmě značně abstrahován, ale součástí práce je i zdrojový kód aplikace, z něhož se dají případné implementační detaily poměrně snadno vyčíst. Na DVD je přiložen i vygenerovaný seznam všech použitých metod včetně jejich typů a parametrů (soubor *BLL.xml*). Kompletní vývojový diagram algoritmu Apriori je pak uveden v příloze (viz. obrázek B.1).

7.2.4 Prezentační vrstva – Presentation Layer

Budeme-li na prezentační vrstvu pohlížet z implementačního hlediska, tak se jedná o projekt typu WinForms, který využívá služeb nižších vrstev. Cílem této vrstvy je i interakce s uživatelem, a to prostřednictvím zakomponováno grafické uživatelské rozhraní. Je tedy žádoucí, aby bylo GUI uživatelsky přívětivé a jeho ovládání bylo intuitivní. Celý vzhled aplikace je koncipován tak, aby byl co nejvíce podobný běžným aplikacím pod operačním systémem MS Windows.

GUI je tvořeno horizontálním menu, levým ovládacím panelem a z velké části také tabulkou, do níž jsou zapisovány vydolované informace. V nejnižší části je umístěn informační panel a v případě probíhajícího procesu je uživateli zobrazen tzv. progress bar. Ošetřeny jsou samozřejmě i všechny vstupy zadávané uživatelem a kontrolována je i posloupnost akcí, které smí uživatel v daném pořadí vykonat (např. nelze spustit proces dolování, pokud nebyla zvolena dolovací metoda, nebo nebyl zadán vlastní dotaz typu DMX).

Jakmile uživatel vybere typ dolovací metody a spustí proces dolování, tak se předává index vybrané dolovací úlohy metodě *btnStartMining_Click()*. Vzápětí se vytvoří nové vlákno, v němž je předáno řízení aplikační vrstvě, která se postará o celý výpočet. Následuje provedení akcí, při nichž se skryjí určité komponenty v GUI, jako například skrytí tlačítka *Spustit proces dolování* nebo skrytí informace, že tabulka neobsahuje žádná data. Po dokončení procesu dolování nastává několik variant, které jsou řešeny v metodě *bwMining_RunWorkerCompleted()*. Buď skončil proces dolování s chybou, nebo zadaným kritériím neodpovídají žádná data (nepodařilo se vydolovat žádné znalosti), nebo jsou data vložena do tabulky a zobrazena uživateli. Pro všechny uvedené případy je uživateli podána přesná informace, jak proces skončil. Pokud uživatel není s výsledkem dolování spokojen nebo již data nepotřebuje mít zobrazená, může tabulku smazat tak, že prostřednictvím tlačítka *Vyčistit data z tabulky* vyvolá metodu *btnClearTable_Click()*.

V následujících částech bude blíže popsána jednak práce s vlákny a na závěr této kapitoly budou detailněji představeny implementované funkcionality, které jsou uvedeny v neformální specifikaci požadavků (viz. kapitola 6). Výsledná podoba grafického uživatelského rozhraní aplikace Data Miner 2011 je znázorněna na obrázku A.1.

7.2.5 Práce s vlákny

Jelikož ze specifikace požadavků plyne mimo jiné i potřeba paralelismu mezi probíhajícími procesy, například mezi procesem dolování a používáním GUI ze strany uživatele, probíhají veškeré výpočty a operace v odděleném vlákně. Tímto je zajištěno, že při výpočtech nedochází k tzv. „vytuhnutí“, či „zamrznutí“ grafického rozhraní.

Pro práci s vlákny poskytuje .NET Framework komponentu *BackgroundWorker*. Vytvoření nového vlákna uskutečníme vyvoláním metody *RunWorkerAsync()*, která způsobí událost *DoWork* a následné spuštění asynchronní úlohy. Chceme-li sledovat průběh výpočtů v rámci asynchronní úlohy, tak je nezbytné nastavit vlastnost *WorkerReportsProgress* na hodnotu *true*. Asynchronní úloha pak poskytuje informace o výpočtech prostřednictvím metody *ReportProgress()*, která vyvolá událost *ProgressChanged*. Pokud si je programátor vědom skutečnosti, že danou asynchronní úlohu bude chtít v jejím průběhu předčasně ukončovat, musí být vlastnost *WorkerSupportsCancellation* nastavena na hodnotu *true*. Pak lze úlohu ukončit tak, že zkontrolujeme, zda-li vlastnost *IsBusy* vrací hodnotu *true* (značí, že úloha skutečně běží) a vzápětí voláme metodu *CancelAsync()*, čímž dojde k přerušení asynchronně vykonávaných operací. V konečné fázi je vždy, ať už byla úloha přerušena či nikoliv, vyvolána událost *RunWorkerCompleted*.

Podrobnější informace o práci s vlákny včetně ukázkových příkladů, nejenom s využitím komponenty *BackgroundWorker*, jsou k dispozici v literatuře [7].

7.2.6 Přehled funkcionalit implementovaných v aplikaci Data Miner 2011

Všemi funkcionalitami, které jsou požadovány v kapitole 6, aplikace Data Miner 2011 disponuje. Výše je popsána práce s vlákny, se kterou úzce souvisí implementovaná funkcionalita **přerušení/zastavení** probíhajícího **procesu dolování**. Jelikož proces dolování probíhá v odděleném vlákně, je pro přerušení výpočtu kontrolována vlastnost *IsBusy* a následně volána metoda *CancelAsync()*. Celý proces přerušení obsluhuje metoda *btnStopMining_Click()*. Uživateli je pro informaci o ukončení procesu zobrazena patřičná informace a dolovací úlohu může opakovaně spustit.

Po vydolování znalostí a jejich následném zobrazení do tabulky má uživatel možnost prostřednictvím tlačítka tyto informace **exportovat do** jednoduchého souborového **formátu CSV**. Implementace exportu byla velmi jednoduchá a přitom je tato funkcionalita poměrně užitečná a smysluplná. V metodě *ExportDataEngine()* se postupně prochází všechny řádky a sloupce a hodnoty se zapisují ve formátu CSV do souboru. Následně je uživateli zobrazen dialog, ve kterém si zvolí název souboru a také adresář pro uložení. Další možností, jak může uživatel s daty zobrazenými v tabulce naložit, je jejich smazání.

Jelikož jsou všechny dolovací dotazy v aplikaci předdefinovány, což by uživatele mohlo určitým způsobem omezovat, aplikace poskytuje grafické rozhraní, s jehož využitím **lze odesílat libovolný DMX dotaz** na Analysis Server. Uživatel znalý jazyka DMX si dokonce může vytvořit i model vlastní a posílat si na tento nový model vytvořené DMX dotazy. Rozpracovaný **DMX dotaz si uživatel může ukládat i načítat**. Pro ukládání i načítání rozpracovaných DMX dotazů je použit **formát XML**. Ukládání dotazu obsluhuje metoda *btnSaveDmx_Click()* ve třídě *DmxQueryForm.cs*, a to opět prostřednictvím ukládacího dialogu, kde je možnost volby názvu i umístění souboru. Načtení dotazu zajišťuje metoda *btnOpenDmx_Click()*, která nabídne uživateli dialog, v němž si vyhledá XML soubor s uloženým dotazem. Dotaz je načten do formuláře a uživatel může pokračovat v jeho úpravách. Funkcionalita tvorby vlastních DMX dotazů a jejich ukládání i načítání je považována za jednu z nejdůležitějších funkcionalit aplikace.

Mnohdy není zobrazování vydolovaných informací v textové či tabulkové formě pro koncové uživatele ideální, a tak je z tohoto důvodu implementována funkcionalita **zobrazení dat do grafu**. Aby uživatel nebyl žádným způsobem limitován, tak si do grafu může nechat vyobrazit libovolný sloupec. Dále je také počítáno se skutečností, že ne všechny řádky vybraných sloupců budou v grafu žádoucí, a tak si uživatel může konkretizovat i rozsahy. Může například zadat, že chce na osu X zobrazit řádky ze sloupce *Age* a na osu Y řádky ze sloupce *Expression*, přičemž rozsah bude desátý až dvacátý řádek. Celá práce s grafy je implementována ve třídě *ChartForm.cs*.

Uživatelská nápověda (manuál) požadovaná v kapitole 6 je řešena formou vyvolání a zobrazení dokumentu *HELP.pdf*, ve kterém je popsána práce s aplikací Data Miner 2011. Zobrazení nápovědy je realizováno kódem `System.Diagnostics.Process.Start(path)`, kde parametr *path* představuje cestu k souboru s nápovědou. Pokud otevření nápovědy skončí neúspěchem (soubor se například nepodařilo najít), je vyvolána výjimka a uživatel je o této skutečnosti odpovídajícím způsobem informován.

Kapitola 8

Testování a navrhovaná rozšíření aplikace

Po dokončení implementační činnosti je potřeba ověřit, zda-li aplikace nejeví známky nestabilního chování a dokáže korektním způsobem reagovat na věškeré vstupy (i nesprávné) a prováděné operace ze strany uživatele. Testování aplikace probíhalo paralelně s vývojem jednotlivých funkcionalit a manuální testy byly prováděny vždy po dokončení implementace rozsáhlejšího modulu či po uvolnění nové verze aplikace.

Testování bylo zaměřeno i na odezvu a rychlost zobrazování vydolovaných informací. U dolovacích úloh, které jsou postaveny nad algoritmy zabudovanými v rámci MS SQL Serveru 2008, konkrétně se jedná o *MS Decision Trees*, *MS Clustering*, *MS Naive Bayes*, trvá proces dolování poměrně krátkou dobu. U algoritmu *Apriori* závisí rychlost výpočtu na zadaných vstupních parametrech (minimální podpora a minimální spolehlivost) ze strany uživatele. Pokud jsou zadány velmi nízké hodnoty, například v okolí jednoho procenta, tak je doba výpočtu v mnoha případech delší. Musíme si však uvědomit, že již princip algoritmu Apriori je výpočetně složitý. Dalším důvodem je, že tabulka *SalesOrderDetail* v databázi *AdventureWorks2008*, se kterou byla většina testů realizována obsahuje 121 317 položek nad nimiž algoritmus provádí operaci *JOIN*. Výsledky testů mohou být poněkud zavádějící z toho důvodu, že jak serverová, tak klientská část byly umístěny na jednom stroji o následující konfiguraci:

- Operační systém: Windows 7 Professional Service Pack 1,
- Typ systému: 32bitový operační systém,
- Procesor: AMD Turion X2 Dual-Core Mobile RM-70 2.00 GHz,
- Operační paměť: 2.00 GB.

Vyšší rychlost výpočtů a lepší odezvu by jistě zaručilo umístění serverové části na výkonný server.

Grafické uživatelské rozhraní aplikace je vytvořeno a stylováno tak, aby jej němel problém obsluhovat i běžný uživatel operačního systému MS Windows. Za tímto účelem byla uživatelům aplikace předložena k „otestování“ a následně měli možnost sdělit své poznatky, připomínky a další zpětnou vazbu prostřednictvím anonymního webového formuláře. Toto rozhodnutí bylo velkým přínosem, neboť jasně ukázalo, že programátor, jakožto tvůrce aplikace si spoustu uživatelských věcí neuvědomuje.

Ze zpětné vazby od uživatelů a z dalších jiných konzultací vyplynulo několik návrhů na vylepšení aplikace. Jedním z hlavních problémů se stalo zobrazování dat do grafu. V původním řešení totiž uživatelé neměli možnost volby sloupců ani volbu rozsahu zobrazených řádků. Na základě těchto poznatků byla funkcionalita doimplementována. Dále pak uživatele omezovala doba průběhu procesu dolování, když nebyla implementována možnost předčasného ukončení takového procesu uživatelem. Ani doimplementování této funkcionality nebylo nijak zvláště problematické. Naopak ocenění sklidila podpora ukládání vydolovaných informací do formátu CSV a také možnost vytvářet, ukládat, načítat a odesílat vlastní DMX dotazy.

Jako případné rozšíření aplikace Data Miner 2011 do budoucna připadá v úvahu implementace grafického uživatelského rozhraní pro tvorbu nových data miningových modelů, včetně definování datových zdrojů a pohledů. V aktuální verzi aplikace mohou uživatelé modely vytvářet jen pomocí vlastních dotazů v jazyce DMX, což je pro mnohé z nich značné omezení, jelikož jazyk DMX neznají. GUI by bylo vhodné implementovat i pro algoritmus *Apriori*, aby si uživatel zvolil, na jaké transakční databázi hodlá analytické výpočty provádět. Tím by se ovšem velká část bakalářské práce zaměřovala spíše na tvorbu GUI, než na problematiku získávání znalostí z databází. Dalším možným rozšířením je do aplikace zakomponovat více dolovacích algoritmů, například výkonný algoritmus *FP-tree*. Lépe propracovat by se dala i operace s grafy, a to jejich ukládáním do různých formátů, nebo pokročilejšími funkcemi pro zobrazování dat (více dimenzí, možnost srovnání více hodnot, atd.). Všechna výše navrhovaná rozšíření je možné do aplikace zakomponovat v některém z budoucích navazujících projektů.

Kapitola 9

Závěr

V první polovině této bakalářské práce je zaveden pojem získávání znalostí z databází a další pojmy s ním související. Vzápětí jsou vysvětleny teoretické principy dolovacích úloh a algoritmů. Struktura teoretické části práce je záměrně koncipována tak, aby byla daná problematika lehce pochopitelná i pro čtenáře, který s oblastí získávání znalostí z databází nemá velké zkušenosti.

Výsledkem praktické části bakalářské práce je desktopová aplikace implementována v jazyce C#, která je postavena na bázi teoretických poznatků obsažených v první části. Aplikace umožňuje uživateli dolování znalostí z databází prostřednictvím vlastní metody Apriori, ale také s využitím metod zabudovaných v rámci MS SQL Serveru 2008. Ve finální verzi splňuje aplikace všechny požadavky vyplývající jednak ze zadání práce, ale i z připomínek a návrhů ze strany uživatelů.

Za velký přínos této práce lze považovat proniknutí do oblasti získávání znalostí z databází a také poznání platformy MS SQL Server 2008, jakožto nástroje pro dolování z dat. Velmi pozitivní je i zkušenost s tvorbou rozsáhlých desktopových databázových aplikací nad platformou .NET Framework verze 4.0. Pochopení objektově orientovaného přístupu a vícevrstvé architektury aplikací lze taktéž považovat za klíčovou znalost pro vývoj dalších aplikací.

Jak již bylo řečeno v předcházející kapitole, případných rozšíření aplikace Data Miner 2011 do budoucna je celá řada. Do aplikace by bylo vhodné zakomponovat i další dolovací úlohy a algoritmy (například algoritmus *FP-tree*). Dále připadá v úvahu rozšíření a propracování grafického uživatelského rozhraní takovým způsobem, aby byla uživatelům umožněna volba zdrojové databáze, nad níž bude dolovací úloha probíhat a aby měl uživatel možnost skrze grafické rozhraní vytvářet nové dolovací modely, včetně jejich datových zdrojů a pohledů.

Literatura

[1] Agarwal, V. V.; Huddleston, J.: *Databáze v C# 2008 : průvodce programátora*. Computer Press, a.s., 2009, iSBN 978-802-5123-096.

[2] Berka, P.: *Dobývání znalostí z databází*. Academia, 2003, iSBN 80-200-1062-9.

[3] Chen, C.-C.; Chen, A.-P.: Using data mining technology to provide a recommendation service in the digital library. *Electronic Library, The*, ročník 25, č. 6, 2007: s. 711–724, ISSN 0264-0473, doi:10.1108/02640470710837137.

[4] Han, J.; Kamber, M.: *Data Mining – Concepts and Techniques, 2nd Edition*. Morgan Kaufmann Publishers, 2006, iSBN 15-586-0901-6.

[5] Hotek, M.: *Microsoft SQL Server 2008 : krok za krokem*. Computer Press, a.s., 2009, iSBN 978-802-5124-666.

[6] Lacko, L.: *Business Intelligence v SQL Serveru 2008 : reportovací, analytické a další datové služby*. Computer Press, a.s., 2009, iSBN 978-80-251-2887-9.

[7] Nagel, C.; Evjen, B.; Glynn, J.; aj.: *C# 2008 : programujeme profesionálně*. Computer Press, a.s., 2009, iSBN 978-802-5123-096.

[8] Rae, R.; Farmer, D.; Walters, R. E.; aj.: *Mistrovství v Microsoft SQL Server 2008 : kompletní průvodce databázového experta*. Computer Press, a.s., 2009, iSBN 978-802-5123-294.

[9] WWW stránky: ADOMD.NET Client Programming. URL http://msdn.microsoft.com/en-us/library/ms123477(v=SQL.100).aspx

[10] WWW stránky: Business Analytics and Business Intelligence Software SAS. URL http://www.sas.com/offices/europe/czech/

[11] Zendulka, J.; Bartík, V.; Lukáš, R.; aj.: *Získávání znalostí z databází. Studijní opora*. FIT VUT v Brně, 2009.

Seznam použitých zkratek a symbolů

KDD – Knowledge Discovery in Databases

BI – Business Intelligence

MS – Microsoft

SQL – Structured Query Language

MDX – Multidimensional Expressions

DMX – Data Mining Extension

SEMMA – (Sample – Explore – Manipulate – Model – Asses)

FTP – File Transfer Protocol

VSTA – Visual Studio Tools for Applications

ERP – Enterprise Resource Planning

OLTP – Online Transaction Processing

OLAP – Online Analytical Processing

GUI – Graphical User Interface

BLL – Business Logic Layer

DAL – Data Access Layer

XML – Extensible Markup Language

CSV – Comma Separated Values

Seznam příloh

Dodatek A – Grafické uživatelské rozhraní aplikace Data Miner 2011
Dodatek B – Vývojový diagram algoritmu Apriori
Dodatek C – Obsah přiloženého DVD

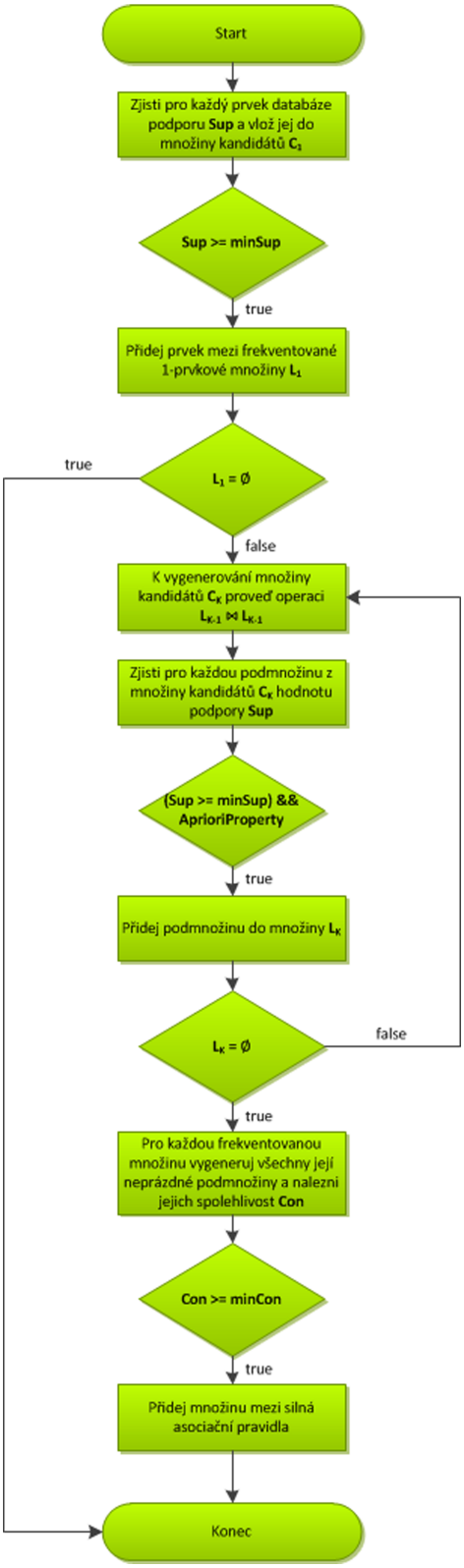
Dodatek A

Grafické uživatelské rozhraní aplikace Data Miner 2011

Obrázek A.1: Grafické uživatelské rozhraní aplikace Data Miner 2011.

Dodatek B

Vývojový diagram algoritmu Apriori



Obrázek B.1: Vývojový diagram algoritmu Apriori.

Dodatek C

Obsah přiloženého DVD

Adresářová struktura přiloženého DVD je následující:

- `Text`
  - `bp_pijacek.pdf` – text bakalářské práce ve formátu .pdf
  - `src` – adresář obsahující zdrojové soubory pro sestavení této bakalářské práce
- `Data Miner 2011`
  - `Client` – adresář obsahující zdrojové soubory pro sestavení klientské desktopové aplikace
  - `Server`
    - `ASDataMiner` – adresář obsahující analytický projekt na straně serveru
    - `DBscripts` – adresář obsahující soubory pro vytvoření zdrojových databází