VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

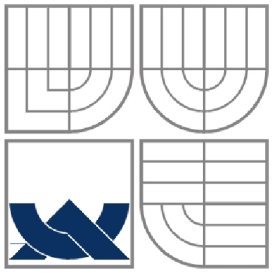
VYTVOŘENÍ NOVÝCH KLASIFIKAČNÍCH MODULŮ V SYSTÉMU PRO DOLOVÁNÍ Z DAT NA PLATFORMĚ NETBEANS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

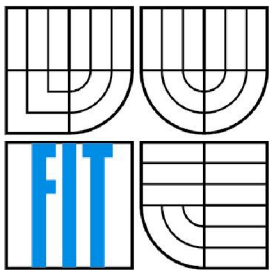
AUTOR PRÁCE
AUTHOR

Bc. Ondřej Kmoščák

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

VYTVOŘENÍ NOVÝCH KLASIFIKAČNÍCH MODULŮ V SYSTÉMU PRO DOLOVÁNÍ Z DAT NA PLATFORMĚ NETBEANS

CREATION OF NEW CLASIFICATION UNITS IN DATA MINING SYSTEM ON NETBEANS PLATFORM

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. Ondřej Kmoščák

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. Roman Lukáš PhD.

BRNO 2009

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2008/2009

Zadání diplomové práce

Řešitel: **Kmoščák Ondřej, Bc.**
Obor: Informační systémy
Téma: **Vytvoření nových klasifikačních modulů v systému pro dolování z dat na platformě NetBeans**
Kategorie: Databáze

Pokyny:

1. Seznamte se s problematikou získávání znalostí z databází. Seznamte se s implementací jádra systému řešeného v diplomové práci Ing. M. Krásného.
2. Po dohodě s vedoucím DP zvolte vhodné dolovací metody z oblasti klasifikace a navrhněte odpovídající dolovací modul pro uvedený systém.
3. Navržený klasifikační modul implementujte.
4. Funkčnost navržených modulů a vlastnosti implementovaných metod ověřte na vhodných reálných datech typických např. pro část informačních sytémů na podporu marketingu či CRM.
5. Zhodnoťte dosažené výsledky a diskutujte další možný vývoj projektu.

Literatura:

- Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Second Edition. Elsevier Inc., 2006, 770 p.
- Krásný, M.: Systém pro dolování z dat v prostředí Oracle. Diplomová práce. FIT VUT v Brně. 2007.

Při obhajobě semestrální části diplomového projektu je požadováno:

- Body 1 a 2.

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

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci ročníkového a semestrálního projektu (30 až 40% 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: **Lukáš Roman, Ing., Ph.D.**, UIFS FIT VUT
Datum zadání: 22. září 2008
Datum odevzdání: 26. května 2009

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Tato diplomová práce se zabývá dolováním z dat a vytvořením dolovacího modulu pro systém pro dolování z dat, který je vyvíjen na FIT. Jedná se o klientskou aplikaci skládající se z jádra a jeho grafického uživatelského rozhraní a nezávislých dolovacích modulů. Aplikace využívá podpory Oracle Data Mining. Systém pro dolování z dat je implementován v jazyce Java a jeho grafické uživatelské rozhraní je postaveno na platformě NetBeans.

Obsahem této práce bude uvedení do problematiky získávání znalostí a následně seznámení s vybranou bayesovskou klasifikační metodou, pro kterou bude následně implementován samostatný dolovací modul. Dále bude popsána implementace tohoto modulu.

Abstract

This diploma thesis deals with the data mining and the creation of data mining unit for data mining system, which is beeing developed at FIT. This is a client application consisting of a kernel and its graphical user interface and independent mining modules. The application uses support of Oracle Data Mining. The data mining system is implemented in Java language and its graphical user interface is built on NetBeans platform.

The content of this work will be the introduction into the issue of knowledge discovery and then the presentation of the chosen Bayesian classification method, for which there will subsequently be implemented the stand-alone data mining module. Furthermore, the implementation of this module will be described.

Klíčová slova

Získávání znalostí z databází, dolování z dat, modul, klasifikace, naivní bayesovská klasifikace, Java, NetBeans.

Keywords

Knowledge discovery in databases, data mining, module, classification, naive bayes classification, Java, NetBeans.

Citace

Kmoščák Ondřej: Vytvoření nových klasifikačních modulů v systému pro dolování z dat na platformě NetBeans, diplomová práce, Brno, FIT VUT v Brně, 2009

Vytvoření nových klasifikačních modulů v systému pro dolování z dat na platformě NetBeans

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Romana Lukáše PhD. Další informace mi poskytli Ing. Michal Krásný a Ing. Jaroslav Ráb. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Ondřej Kmoščák
22.5.2009

Poděkování

V této části bych rád poděkoval vedoucímu mé diplomové práce Ing. Romanu Lukášovi, PhD. za jeho odbornou pomoc, ochotu, rady a věcné připomínky a konzultace.

Obsah

1 Úvod

V současné době jsme se díky vývoji dostali do situace, kdy jsme doslova zaplavováni obrovským množstvím dat. Tato data jsou ovšem opravdu surovými daty, ve kterých se lze jen stěží orientovat. Je tedy nutné těmto datům přiřadit jasnou sémantiku a tím z nich udělat informaci a následně z těchto informací získat nové znalosti, které nejsou doposud známé. Jako první v postupném vývoji přišly na řadu takzvané Datové sklady a spolu s ním technologie OLAP (z angl. On Line Analytical Processing), které oddělovaly data pro analýzu od dat uložených v produkční databázi. Tento způsob analýzy dat probíhá na základě interakce s uživatelem, který danou analýzu provádí. V současnosti převládá trend tento postup pokud možno co nejvíce automatizovat a poté tyto znalosti využívat. Jako reakce na požadavek automatizace celého procesu vzniky systémy pro získávání znalostí z databází. Pole využití takto získaných znalostí tímto automatizovaným procesem je takřka neomezené, jediným limitujícím faktorem v tomto procesu je fakt, že ne ze všech dat je možné vydolovat nějakou novou znalost. Nejprve bylo zamýšleno využití nově nabytých znalostí pomocí těchto technologií pouze v oblasti marketingu. Postupem času ovšem tento proces našel uplatnění i v dalších oborech, jako jsou zdravotnictví a medicína, pojišťovnictví, bankovnictví, průmysl a v řadě dalších.

Nyní si vymezme pojem „získávání znalostí z databází“. Pod tímto pojmem si můžeme představit extrakci užitečných a zajímavých informací, které jsou netriviální, tudíž je nelze získat pomocí pouhého dotazovaní pomocí SQL, nepatří sem ani deduktivní databáze a expertní systémy. Tyto informace také musí být skryté, neboli hledáme modely a vzory, které nejsou na první pohled patrné, stejně tak tyto informace musí být dříve neznámé a potenciálně chtěné [Zendulka]. Neboli získávání znalostí z databází je získávání dříve neznámých na první pohled nezřejmých informací, které nám mohou napomoci například v našem dalším rozhodování.

Tato diplomová práce se zabývá problematikou získávání znalostí z databází a systémem pro dolování z dat, který je v současné době vyvíjen na FIT, a jeho následným rozšířením o dolovací modul z oblasti klasifikace. Tento diplomový projekt navazuje a čerpá ze semestrálního projektu [Kmoščák] a navazuje na práci Ing. Krásného [Krásný] a Ing. Madera [Mader], kteří se tímto systémem ve svých diplomových projektech již zabývali. Celý projekt je implementován v jazyce Java a je založen na platformě NetBeans. Systém spolupracuje s relační databází od firmy Oracle, proto je také využíváno podpory pro dolování z dat Oracle Data Mining, pro implementaci dolovacích modulů pro tento systém. Podrobněji se s tímto systémem seznámíme později.

V následujících kapitolách se seznámíme s problematikou získávání znalostí z databází jako takovou. Budou zmíněny použité technologie a také bude podrobně představen systém pro dolování z dat, který je vyvíjen na FIT. Následně bude uvedena problematika tvorby dolovacího modu pro tento systém, stejně tak bude diskutována implementace konkrétního dolovacího modulu používající naivní bayesovskou klasifikaci. V závěru práce nalezneme zhodnocení celého procesu tvorby dolovacího modulu, ukázkový příklad a možná vylepšení do budoucna.

2 Získávání znalostí z databází

V této kapitole se seznámíme s historií a vývojem získávání znalostní z databází, popíšeme si podrobněji celý proces získávání znalostí a jeho jednotlivé fáze. Budou diskutovány i základní typy dolovacích úloh a zmíníme se o vhodných datech pro dolování.

2.1 Historie a vývoj

Počátek databázových systémů a tedy systému pro uchování perzistentních dat a jejich manipulaci se datuje do 60. let dvacátého století. Toto období bylo obdobím vzniku databázových technologií jako takových. Z počátku byly používány dva druhy databázových modelů – hierarchický a síťový, který byl jakýmsi zobecněním síťového modelu. K velkému posunu došlo v roce 1970, kdy Edgar Frank Codd pracující pro firmu IBM položil základy teorie relačního modelu databáze, který dodnes zůstává tím nejpoužívanějším a nejúspěšnějším modelem. V 80. létech se upevnilo postavení relačního modelu se současnou snahou poskytnout modely pro složitě strukturovaná data, či pro specifickou práci s daty – například deduktivní model, či objektově orientovaný model. Dále se začaly objevovat aplikačně orientované SŘBD pro prostorové, temporální a jiné databáze [Zendulka].

Tento obrovský rozmach databázových technologií vedl k tomu, že se v 90. letech začaly objevovat nové nástroje a technologie pro analýzu takto uložených dat. Stejně tak se začaly objevovat technologie pro tvorby datových skladišť, sloužících ke shromažďování a přípravu dat z různých zdrojů pro následnou analýzu. Datové sklady tedy nepodporují pouze uchovávání dat, ale umožňují i jejich analýzu. Model dat v takovém skladu má podobu vícedimenzionální kostky, nad kterou jsou definované určité operace. Taková oddělená analýza se nazývá OLAP (*online analytic procesing*).

V 90. letech se kromě OLAP začíná objevovat jako reakce na silnou potřebu analyzovat velké objemy dat právě získávání znalostí z databází. Rozdíl mezi OLAP a získáváním znalostí je v tom, že na rozdíl od OLAP operací, které uživatel používá interaktivně, je u získávání znalostí z databází snaha o automatizaci nalezení potencionálně chtěných vzorů a modelů v datech. Toto období je také obdobím obrovského rozmachu internetu, na který lze také pohlížen jako na distribuovanou heterogenní databázi obsahující nejrůznější data.

Počátek 21. století je obdobím rozvoje metod a nástrojů pro získávání znalostí z databází a jejich aplikace v praxi. Stejně tak ale vznikají i nové možnosti a podněty k dalšímu výzkumu. Setkáváme se například s pojmy jako dolování v proudech dat. Proudem dat rozumíme data souvisle vznikající ve zdroji, které je třeba nějak spravovat a analyzovat a to velice často online. Takovým proudem dat může být telefonní hovor, záznam z bezpečnostní kamery, ale i řada dalších.

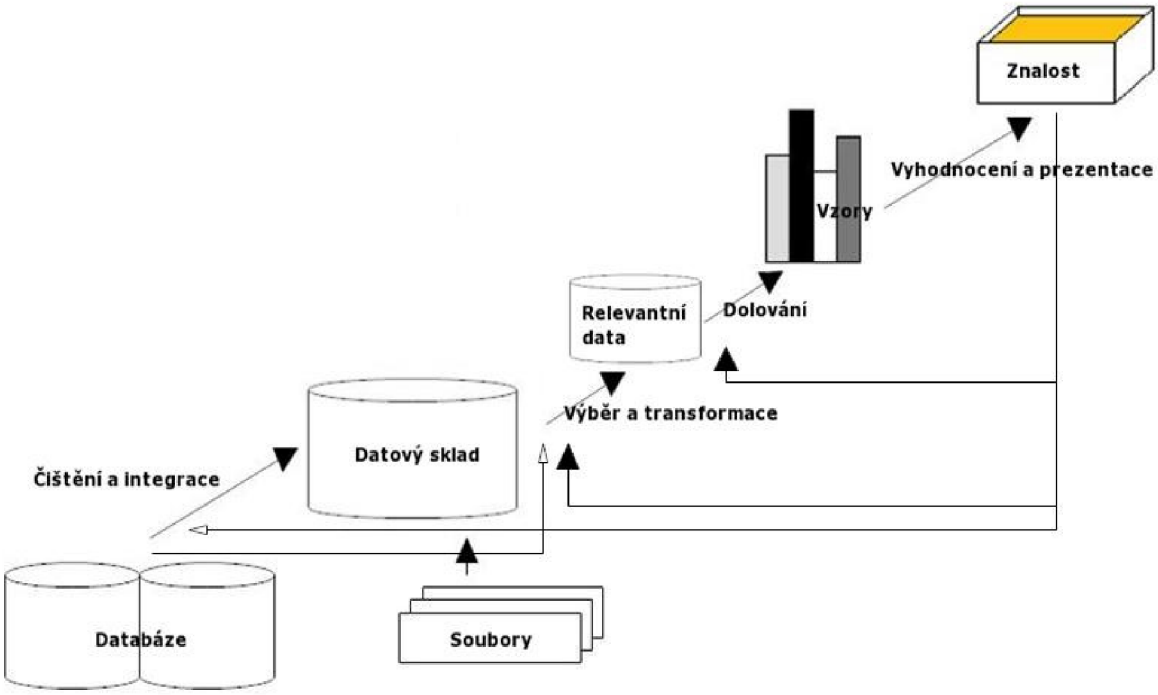
2.2 Proces získávání znalostí z databází

Proces získávání znalostí je znázorněn na Obrázku 2.2.1. Dále si projdeme jednotlivé fáze tohoto procesu a podrobněji je popíšeme. Jak je z obrázku patrné, některé kroky se mohou opakovat. Celý proces se pak skládá z následujících sedmi kroků:

1. ***Čištění dat***

   Klasická data mohou obsahovat šum, chyby, nemusí být úplná, nebo mohou být v nekonzistentním stavu. Tato fáze se právě zabývá odstraněním těchto chyb a nedostatků. Chybějící data můžeme doplnit buďto z jiných zdrojů, nebo odhadnout na základě jiných údajů,

chybná a nekonzistentní data můžeme odstranit, nebo opět získat z jiného zdroje, máme-li tuto možnost.



Obrázek 2.2.1 *Proces získávání znalostí z databází* – přepracováno ze [Zendulka]

2. ***Integrace dat***

   Data, ze kterých chceme znalosti získávat, nejsou vždy pouze v jedné databázi. Většinou data pochází z několika různých databází, případně souborů, proto je velmi vhodné ukládat tato data do *datových skladů*. Integrace dat se zpravidla provádí spolu s čištěním dat, jelikož jedním z hlavních důvodů nekonzistence je právě to, že data pochází od různých zdrojů. Ukládání do datových skladů ovšem není nutnou podmínkou, z dat lze dolovat i bez jeho použití (viz obrázek 2.2.1), ovšem zpravidla se datových skladů používá, ať už z důvodu zmíněného výše, tak i z důvodu, že poskytuje řadu dalších analytických služeb.

3. ***Výběr dat***

   V tomto kroku je úkolem vybrat ta data, která jsou pro naši dolovací úlohu relevantní. Pracujeme-li s relační databází, jedná se klasicky o výběr vhodné tabulky a z ní pak vhodných sloupců (atributů). Při použití datových skladišť je úkolem volba vhodné dimenze.

4. ***Transformace dat***

   Pro dolování je nutné data převést do vhodné podoby. Pod touto transformací si můžeme představit například *agregaci*, či *sumarizaci*. V případě, že je použit datový sklad, je možné zařadit tento krok před výběr dat, jelikož transformace dat může být již součástí tvorby datového skladu.

5. ***Dolování dat***

   Toto je stěžejní krok celého procesu a je jádrem získávání znalostí z databází. Během procesu dolování získáváme případné *vzory* (asociační pravidla), či *modely* (klasifikace) pomocí určitých metod.

6. ***Hodnocení modelů a vzorů***

   V tomto kroku je snaha vybrat ty nejzajímavější a nejužitečnější z obrovské řady vzorů, které se v databázi nacházejí. K tomu se používá tzv. *míry užitečnosti*. Užitečnost je měřena významem pro nějaké rozhodnutí – typicky tímto rozhodnutím bývá rozhodnutí banky, zda danému klientovi poskytnout úvěr, či nikoli.

7. ***Prezentace výsledků***

   V posledním kroku je zapotřebí prezentovat dosažené výsledky uživateli za pomocí technik vizualizace a prezentace znalostí. Tato část je neméně důležitá než kroky předešlé. Je nutné, aby i uživatel bez odborného vzdělání a bez podrobné orientace v problematice dolování z dat pochopil, co za znalosti bylo získáno. Bez kvalitní a pochopitelné prezentace by všechny předchozí kroky přišly nazmar.

Na tomto místě bych ještě upozornil čtenáře na fakt, že i když se často setkáváme s pojmem „dolování dat“ z anglického „data mining“, jde vlastně o chybné pojmenování. Protože my nedolujeme data, ale snažíme se dolovat znalosti z dat. Také je nutno říci, že samotné dolování je pouze jedním z kroků, které při získávání znalostí z databází provádíme, přesto se v praxi berou pojmy získávání znalostí z databází a dolování dat jako synonyma a tak je budeme chápat i v následujícím textu této práce.

2.3 Příprava dat pro dolování

V reálném životě se setkáváme s databázemi, které zpravidla nejsou v takovém stavu, abychom data, která se v nich nacházejí, bezprostředně postoupit dolovacímu algoritmu k dolování. Víme, že často obsahují data, která jsou zašuměná, nekonzistentní a můžeme se setkat i s tím, že některé hodnoty zcela chybí. Takovým datům říkáme *nečistá* (případně *dirty*). Tyto záporné vlastnosti vstupních dat by mohly vést k nepřesným nebo dokonce nesprávným a zavádějícím závěrům vyplývajícím z vydolovaných znalostí [Zendulka].

Data v surové podobě, ve které se obvykle nacházejí v databázích, tedy nejsou vhodné pro dolování a je nutné je před samotným dolováním předpřipravit. Tomuto předzpracovaní a přípravě se budeme věnovat právě v této kapitole.

2.3.1 Čištění dat

Čištění dat má za úkol vyřešit problém chybějících hodnot, nekonzistenci dat a odstranit odlehlé hodnoty, které by mohly mít negativní vliv na výsledek dolování. Dále by mělo zajistit vyhlazení dat a odstranění redundance. Čištění dat je proces skládající se ze dvou kroků [Zendulka].

1. **Detekce nesrovnalostí**

   V tomto kroku jsou velice důležitá metadata. Slouží k pochopení dat jako celku a k následnému nalezení nesrovnalostí. Neboli čím více informací o datech máme k dispozici, tím vetší šanci máme daná data pochopit a nalézt v nich případné nesrovnalosti. Dále lze také s výhodou použít tzv. sumarizujících charakteristik. Častými zdroji nekonzistence je použití různých formátů. Například u data lze použít celou řadu formátů a to následně může vést k nekonzistenci. Velice vhodná je také kontrola unikátních hodnot, zda jsou opravdu unikátní. Obecné nástroje pro automatickou detekci nesrovnalostí jsou takřka nedostupné. Spíše se můžeme setkat s nástroji specializovanými jako je například nástroj pro kontrolu poštovního směrovacího čísla a tak podobně.

2. **Odstranění nesrovnalostí**

   Je nutno říci, že některé nesrovnalosti lze odstranit ručně, ale ve valné většině případů je třeba automatizovaného postupu, který nesrovnalosti odstraní pomocí transformací. Existují dostupné nástroje, jako jsou například nástroje pro migraci dat, které provádí jednoduché transformace, jako je náhrada hodnot atributu známka z 1 na 'A' apod. Větší možnosti potom nabízejí nástroje *ETL* (*Extraction/Transformation/Loading*).

2.3.1.1 Odstranění chybějících hodnot

Tento negativní jev můžeme vyřešit několika způsoby. Nejjednodušší způsob řešení je **ignorování celé n-tice**. V takovém případě ale přicházíme o hodnoty ostatních atributů. Ignorováním celých n-tic tedy můžeme přijít i o důležité informace. Toto řešení najde snad jen své použití při klasifikaci, kdy chybí hodnota u cílového atributu. Takový záznam je pro nás zbytečný a můžeme si dovolit jej zcela vynechat, jinak se tato metoda příliš nevyužívá.

Dalším jednoduchým způsobem odstranění chybějících hodnot je **manuální doplnění**. Uživatel může data procházet a doplňovat hodnoty na základě svých znalostí. Tento postup má za předpokladu dobrých znalostí uživatele velice dobré výsledky, ovšem na příliš velké množství dat je z časových důvodu v praxi zcela nepoužitelný.

Poslední možností jak tento problém řešit je **automatická náhrada** chybějících hodnot. Existuje několik variant automatického nahrazení a na ty se nyní podáváme blíže.

Nahrazení globální konstantou

Jde v podstatě o jakousi obdobu hodnoty *NULL*. Jen je nutné, aby šlo o nějakou skutečnou hodnotu atributu, například 0 pro numerický atribut. Bude-li hodnota mimo rozsah, může být následně brána jako šum v datech. Je ovšem nutné takovou hodnotu vhodně zvolit, aby neměla v důsledku negativní vliv na výsledky.

Nahrazení průměrnou hodnotou

Jde o klasické nahrazení hodnotou aritmetického průměru ostatních nechybějících hodnot daného atributu.

Průměrnou hodnotou n-tic patřících do téže třídy

V podstatě se jedná o nahrazení průměrem hodnot daného atributu jen určité podskupiny n-tic. Je možné to ilustrovat na příkladě, kdy chybí hodnota u atributu „cena" u automobilu, u kterého je atribut „třída" rovný hodnotě „střední", bude jeho atribut „cena" doplněn průměrem hodnot atributů „cena" všech automobilů, jejich atribut „třída" je rovný hodnotě „střední".

Nejpravděpodobnější hodnotou

Tato hodnota se určí odhadne pomocí hodnot ostatních atributů. Jedná se vlastně o aplikaci dolovací úlohy při přípravě dat. Používá se *regrese* pro spojité atributy (cena, výše platu), *bayesovská klasifikace*, či *rozhodovací strom* pro atributy diskrétní (schopnost splácet atp.).

2.3.1.2 Odstranění šumu v datech

Pod tímto pojmem rozumíme náhodné chyby v datech, které mohou být způsobeny řadou příčin. Může se jednat o chybu hardware (chyba zařízení při sběru dat), software, ale i o selhání lidského faktoru. Například uživatel zadá do systému věk zákazníka místo 24,0 hodnotu 240. Existuje opět několik způsobu řešení tohoto problému.

Plnění (binning)
Tato metoda provádí lokální *vyhlazení* na setříděných numerických datech. Neboli zohledňuje hodnoty v blízkém okolí. Tedy data jsou nejprve rozdělena do *košů* (*bins*) tak, že v každém koši je přibližně stejný počet hodnot. Poté se jednotlivé hodnoty nahradí buď průměrem koše, mediánem koše, nebo maximem či minimem koše dle toho, ke kterému má daná hodnota blíže.

Regrese
Data jsou nahrazena hodnotami, které jsou dány regresní křivkou. U lineární regrese se nalezne přímka, která nejlépe aproximuje hodnoty dvou atributů. Vícenásobnou lineární regresi, která je rozšířením lineární regrese, lze použít k predikci více než dvou atributů.

Shlukování
Tato metoda může být účinnou pro detekci odlehlých hodnot. Hodnoty, které po shlukové analýze nepatří do žádného shluku, můžeme označit jako odlehlé a tedy i za šum. Více o této metodě zmíníme v kapitole 2.6.4.

2.3.2 Integrace

Je možné, že se dostaneme do situace, kdy bude nutné spojit data z více zdrojů do jednotného celku, neboli bude nutné provést jejich integraci. V této kapitole si popíšeme hlavní problémy, které spolu s integrací souvisí.

Konflikty schématu
V případě, že máme k dispozici data z více zdrojů, může dojít k tomu, že v některém ze zdrojů může být stejná informace vyjádřena odlišným schématem. Například adresa se v jedné databázi jednoho zdroje může skládat z atributů ulice, číslo, město a PSČ, zatímco v databázi jiného zdroje to může být jeden neatomický atribut. Neboli je nutné integrovat *metadata* více zdrojů do jedněch metadat popisujících výsledný zdroj dat [Zendulka]. K řešení konfliktů schématu je tedy zapotřebí mít dostupná metadata.

Konflikty hodnot
Jako příklad takového konfliktu můžeme uvést, že atribut jméno v databázi jednoho zdroje je uloženo jako „Pepa Novák“, zatím co v druhé je uloženo jako „Josef Novák“. Jde tedy o případ, kdy jsou hodnoty odpovídajících si atributů různé. Příčiny takovýchto konfliktů může být celá řada, například rozdílné stupnice či jednotky (známky, teplota...), odlišné konvence atp. Pro vyřešení takovýchto konfliktů je nutný převod do jednotné formy.

Konflikty identifikace
Jedná se o problém, že stejný objekt reálného světa může být ve dvou různých databázích identifikován rozdílně nebo různé objekty mohou být identifikovány stejně. Přiřazujeme–li například objektům jednoznačný identifikátor, který je unikátní pouze pro danou databázi, pak při sloučení více takovýchto databází dojde k porušení této jednoznačnosti – více objektů bude mít stejný identifikátor.

Konflikty redundance
Někdy se můžeme setkat s jevem, že se v databázi nacházejí atributy, které lze jednoduše odvodit z atributů jiných. Například z data narození lze jednoduše odvodit věk atp. Taková data jsou pro nás redundantní. Nemusí tedy jít pouze o výskyt totožných atributů. Tento jev může vést ke zcela bezvýznamným výsledkům při dolování.

2.3.3 Transformace dat

Transformace dat má za úkol transformovat data do takového tvaru a podoby, který bude vhodný pro dolování. Transformace může provádět následující operace [Zendulka].

- **Normalizace** – transformace dat do požadovaného intervalu, nejčastěji se jedná o intervaly <0,1> nebo <-1,1>.
- **Generalizace** – zdrojová data jsou nahrazena koncepty z konceptuální hierarchie. Například kategorický atribut „město“ může být na vyšší úrovni nahrazen názvem kraje.
- **Vyhlazení** – jde o odstranění šumu zmiňovaném v kapitole 2.3.1.2.
- **Konstrukce** – z několika různých atributů odvodíme jeden za účelem zefektivnění dolovacího procesu. Například z atributů „daňový základ“ a „daňová sazba“ můžeme vytvořit atribut „výše daně“.
- **Agregace** – podrobné a detailní atributy jsou agregovány.

2.4 Data vhodná pro dolování

Nyní se dostáváme k otázce, z jakých dat je vůbec možné dolovat. V zásadě platí, že dolování dat by mělo být použitelné jak pro *datová skladiště* s persistentními daty, stejně tak pro data proměnlivá, kterými jsou například *datové proudy* [Han]. Pro dolování lze tedy použít relační databáze, datové sklady, transakční databáze, moderní databázové systémy, tzv. „*flat files*“ (soubory popisující tabulku zpravidla jako plain text), *datové proudy* a *World Wide Web*. Mezi pokročilé databázové systémy patří *objektově-relační databáze* a specifické *aplikačně-orientované* databáze, jako jsou prostorové, temporální, multimediální a další databáze.

Systém, který je vyvíjen na FIT, pracuje s *relačními databázemi*. Je to i z důvodu, že v současnosti jsou relační databáze asi nejčastějšími zdroji pro dolování. Proto si podrobněji zmíníme jen tento zdroj dat, ze kterých je možno dolovat.

2.5 Relační databáze

Relační databází rozumíme kolekci spolu souvisejících databázových tabulek, které splňují alespoň *první normální formu*, neboli všechny atributy dané tabulky jsou *atomické*.

Data v databázi jsou pak zpřístupňována pomocí dotazovacího jazyka, kterým je typicky jazyk SQL. Díky takto položeným dotazům můžeme například z tabulky o zákaznících zjistit, kolik zákazníků bydlí v daném městě, nebo jaké zboží si zákazník koupil. Výsledky těchto dotazů lze sice použít pro další rozhodování, ovšem nejedná se v žádném případě o dolování znalostí, jak jsme si jej definovali.

Dolování nám může poskytnout komplexní informace. Při dolování můžeme nalézat *trendy*, *datové vzory* a *modely*. Například pomocí dolování mohou systémy analyzovat údaje o zákaznících, aby předpověděly míru úvěrového rizika u nových zákazníků vycházejícího z jejich příjmů, věku a předchozích úvěrů. Dolovací systémy mohou také zjistit odchylky v prodeji daného zboží od očekávaného prodeje v porovnání s předešlými roky. Tyto odchylky pak lze podrobněji zkoumat a analyzovat, jestli například nedošlo ke změně balení nebo výrazné změně ceny.

Jak již bylo zmíněno dříve, relační databáze jsou jedním z nejběžnějších a nejdostupnějších zdrojů dat pro dolování.

2.6 Typy dolovacích úloh

V této části práce bych zmínil jednotlivý typy dolovacích úloh. V literatuře je to co nazýváme typem dolovací úlohy nazýváno funkcionalita dolování dat (z angl. Data mining functionality) [Zendulka]. Jde tedy o to, jaký druh nebo typ modelu dat se snažíme dolováním z dat získat.

Typy úloh můžeme rozdělit do dvou následujících skupin:

1. **Deskriptivní**
   Deskriptivní úloha charakterizuje obecné charakteristiky zkoumaných dat na základě údajů v databázi. Nejznámějším případem je zjišťování společného nákupu různých druhů zboží při analýze nákupního košíku, zjišťovaného managementem prodejců. Například zjistíme-li, že zákazník kupuje spolu s výrobkem A i výrobek B, umístíme podle toho tyto výrobky na prodejní ploše.
2. **Prediktivní**
   Na základě analýzy současných dat se snaží najít heuristiky pro předpověď budoucího chování. Příkladem je například klasifikace (pro předpověď nespojitých vlastností) a predikce (pro předpověď spojitých atributů). Příkladem použití klasifikace je již případ, kdy se snažíme zjistit míru rizik, pro poskytnutí úvěru.

Současné dolovací systémy poskytují možnosti řešení různých dolovacích úloh a také podporují řadu algoritmů pro jejich řešení, a to ze dvou důvodů. Zaprvé se snaží být dostatečně univerzálními a zadruhé se snaží vyrovnat s faktem, že uživatelé nemají dostatečnou představu a znalosti k tomu, aby věděli, jaký model jejich dat může být právě pro ně užitečný.

Nyní si podrobněji popíšeme některé typy dolovacích úloh.

2.6.1 Popis konceptu/třídy

Neboli concept/class description je jedním ze základních typů dolovacích úloh. V tomto případě využíváme toho, že data mohou být asociována s určitým konceptem, či třídou. Například v obchodě mohou existovat třídy položek počítač a tiskárna týkající se zboží a mohou být zavedeny pojmy utráceč a rozpočtář náležící k zákazníkům. Poté je vhodné stručně a souhrnně popsat koncepty dostatečně přesným způsobem. Takovýto způsob, který budeme nazývat popisem konceptu/třídy, lze získat jedním ze dvou následujících popisů.

Prvním z nich je *charakterizace dat*, ta značí souhrn obecných vlastností analyzované třídy. Data odpovídající dané třídě je možné z databáze získat pomocí jednoduchého dotazu. Například třídou by mohly být počítače, jejichž prodej za minulý měsíc stoupl o 20%. Existuje několik metod pro charakterizaci a sumarizaci dat. Například OLAP operace roll-up nad datovou kostkou, nebo automatizovaná indukce orientovaná na atributy.

Druhým způsobem je tzv. *diskriminace dat* , ta se od charakterizace liší tím, že nepopisujeme data analyzované třídy pomocí obecných výrazů, ale vymezujeme je ve vztahu k jedné, či několika rozdílovým třídám. Neboli se snažíme nalézt atributy a jejich hodnoty, ve kterých se nejvíce liší. Takovou úlohu si může přestavit jako dotaz: „Čím se liší produkty, jejichž prodej loni klesl minimálně o 15 procent od produktů, jejichž prodej loni vzrostl alespoň o 25%?“.

2.6.2 Frekventované vzory, asociace a korelace

Dalšími typy úloh jsou úlohy pro odhalovaní vztahů mezi atributy. Mezi ně patří dolování frekventovaných vzorů, asociací a korelací.

Frekventované vzory

Jak již název napovídá, jedná se o vzory, které se v analyzovaných datech vyskytují často. Tyto vzory jsme již diskutovali, když jsme pojednávali o analýze nákupního košíku v této kapitole při popisu deskriptivních dolovacích úloh, kde se jednalo o frekventované množiny. V případě frekventovaných posloupností, například že zákazníci mají tendenci koupit nejprve digitální fotoaparát a až poté paměťovou kartu, mluvíme o tzv. *sekvenčních vzorech*.

Dále je možné uvažovat i jiné podstruktury, jako jsou podstromy, podgrafy atp. Jestli se v datech představujících kolekci struktur vyskytuje často, hovoříme pak o tzv. (frekventovaných) *strukturovaných vzorech*.

Příkladem asociační analýzy může být například to, že bychom jako manageři firmy, chtěli zjistit, které z nabízených výrobků se prodávají často a společně. Můžeme například zjistit, že ve 23% všech nákupů se výrobek A prodal společně s výrobkem B. Ve zbylých případech si zákazník nekoupil tyto výrobky dohromady. Dále jsme také zjistili, že v 60% případů, kdy si zákazník koupil výrobek A si také koupil výrobek B. Výsledkem toho zjištění je tzv. asociační pravidlo ve tvaru:

$$koupí(X, \text{'výrobek A'}) \Rightarrow koupí(X, \text{'výrobek B'})[podpora = 23\%, spolehlivost = 60\%]$$

kde X je proměnná značící zákazníka. Toto pravidlo říká, že zákazník, který koupí výrobek A má tendenci koupit i výrobek B. Podpora a spolehlivost jsou pak dvě nečastěji používané míry vyjadřující významnost zastoupení daného frekventovaného vzoru v analyzovaných datech.

Výše uvedené asociační pravidlo obsahuje pouze predikát *koupí*, proto se nazývá jednodimenzionálním asociačním pravidlem. V případě že pravidlo obsahuje více než jeden predikát, nazýváme jej multidimenzionálním asociačním pravidlem. Příklad takového pravidla by mohl být následující:

$$Věk(X, '20 .. 39') \wedge Plat(X, '30K .. 39K') \Rightarrow koupí(X, \text{'B'})[podpora = 10\%, spolehlivost = 50\%],$$

toto pravidlo říká, že 10 procent analyzovaných zákazníků ve věku 20 až 39 let s platem v rozmezí 30 000 až 39 000 si koupilo výrobek B. Je pravděpodobnost 60%, že zákazník ve stejné věkové a platové kategorii si také koupí výrobek B.

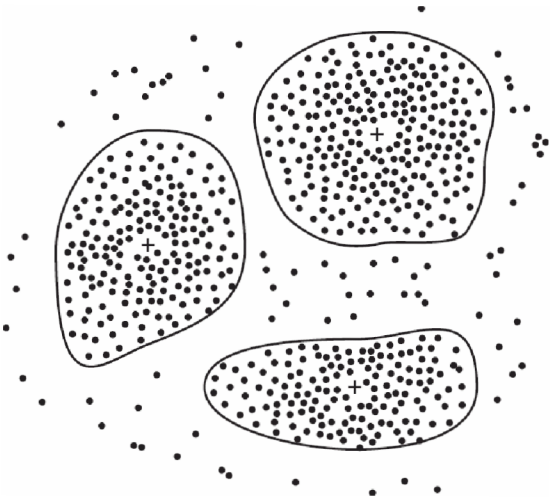
2.6.3 Analýza evoluce

Nyní si popíšeme další typ dolovací úlohy. Analýza evoluce popisuje a modeluje pravidelnosti a trendy u objektů měnících své chování v čase. I když i u dat se vztahem k času lze použít i jiné úlohy, zmíněné v této kapitole, existují speciálně zaměřené úlohy, jako je vyhledávání periodicity, analýza podobností atp.

2.6.4 Shluková analýza

Tento typ dolovací úlohy má za úkol nalézt třídy objektů, které mají co nejvíce společného a naopak se od ostatních tříd co nejvíce liší. Objekty se shlukují do tříd na principu maximalizace podobnosti téže třídy a minimalizace podobnosti objektů různých tříd. Takto vytvořené třídy mají podobu shluků.

Tento způsob analýzy lze použít například na zákazníky firmy s cílem nalézt homogenní skupiny zákazníků, kteří bydlí v blízkosti daných prodejen. Tuto znalost pak lze úspěšně použít pro cílenou reklamní kampaň. Jak jsme si již řekli, někdy nás mohou zajímat takové vzory, které se naopak vyskytují velice zřídka, neboli takové, které se velmi odlišují od ostatních. Takové datové objekty se nazývají odlehlé objekty a jejich hledání se nazývá dolování odlehlých objektů.



Obrázek 2.6.4.1 *Shluková analýza* – převzato ze [Zendulka]

Dalšími velice významnými typy dolovacích úloh jsou klasifikace a predikce. Klasifikace je zásadní problematikou pro tuto diplomovou práci, proto se jí budeme věnovat samostatně v následující kapitole.

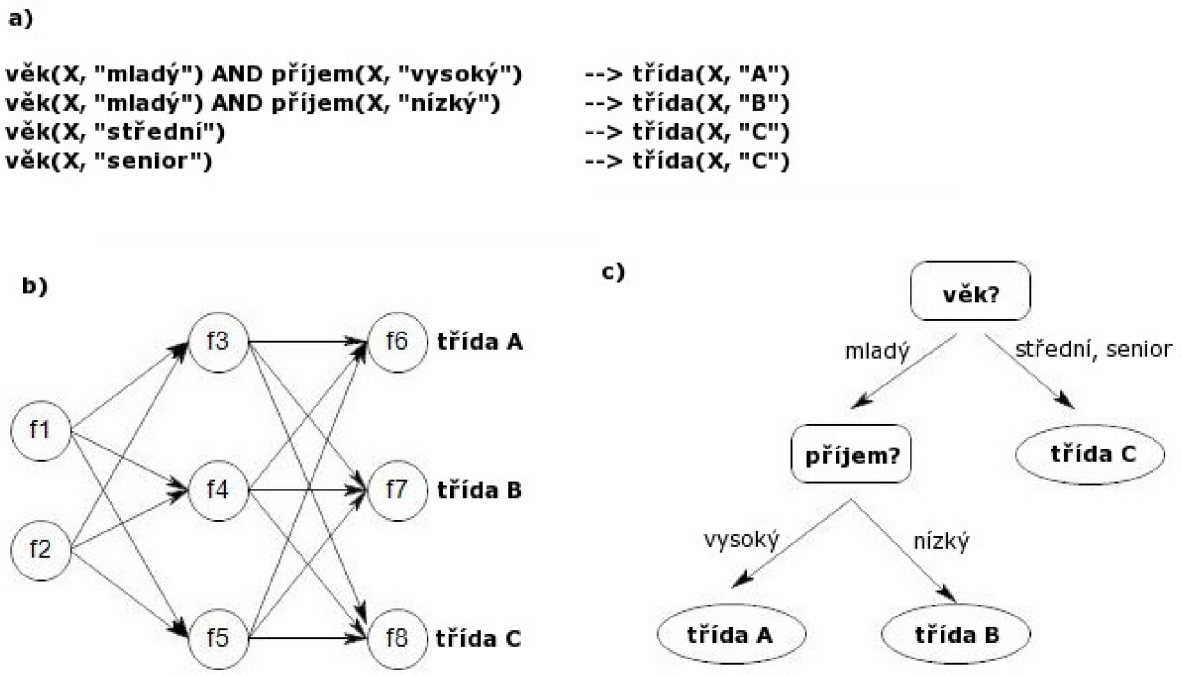
3 Klasifikace a predikce

Oba tyto typy dolovací úlohy patří mezi úlohy prediktivní, jak jsme si je popsali v kapitole 2.6. Nejprve se budeme zabývat klasifikací a poté predikcí. Na závěr si oba typy úloh porovnáme a zmíníme rozdíly.

3.1 Klasifikace

Cílem klasifikace je nalézt takový *model*, který popisuje a současně rozlišuje třídy dat, a ten poté použít pro predikci třídy, do které bude objekt jehož zařazení neznáme patřit [Zendulka]. Neboli obecně je to proces, který umožní data rozdělit na základě jejich vlastností, do daných tříd. Těchto tříd, do kterých klasifikujeme, musí nutně být konečný počet. Je nutné podotknout, že klasifikace se používá k předpovězení diskrétních hodnot, čímž se tedy liší od predikce, která předpovídá pravděpodobnou přesnou hodnotu, která je ze spojitého intervalu. Klasifikační proces probíhá v následujících třech krocích [Zendulka]:

1. **Trénování** – na základě analýzy tzv. trénovací množiny je vytvořen klasifikační model.
2. **Testování** – hodnocení kvality vytvořeného modelu použitím testovacích dat.
3. **Aplikace** – použití modelu pro klasifikaci objektu, jehož třídu neznáme.



Obrázek 3.1.1 *Formy klasifikačního modelu: a) klasifikační pravidla, b) neuronová síť, c) rozhodovací strom* – přepracováno z [Han]

V některých zdrojích se aplikace neuvažuje jako krok část samotného procesu klasifikace, v tom případě se uvažují jen první dva kroky. První fáze je trénovaní (učení) probíhající na trénovací množině a poté následuje fáze testovací, která probíhá na množině testovací. Trénovací a testovací data jsou pouze data objektů, u kterých známe třídu. V případě trénování tuto skutečnost využíváme k vytvoření modelu a v případě testování nám slouží k určení poměru správně a špatně

klasifikovaných objektů tudíž k zjištění jeho přesnosti. Neboli zjistíme, v kolika případech model neklasifikuje správně, díky tomu můžeme rozhodnout, zda je daný model správný a prohlásíme jej za použitelný, nebo jej musíme upravit, případně zjistíme, že je třeba vytvořit model zcela nový.

Model pro klasifikaci může mít různé podoby. Používají se například rozhodovací stromy, klasifikační (IF-THEN) pravidla, matematické formule nebo neuronové sítě. Některé z nich jsou uvedeny na obrázku 3.1.1.

3.1.1 Příprava dat pro klasifikaci

Aby byl proces klasifikace co možná nejefektivnější a klasifikátor dosahoval co nejlepších výsledku je třeba speciálním způsobem surová data připravit. Mezi tyto úpravy patří [Lukáš]:

- **Čištění dat** – odstranění šumu v datech a nahrazení chybějících údajů zpravidla nejčastěji se vyskytující hodnotou, případně vynechání neúplných dat.
- **Významnostní analýza** – odstranění atributů, které jsou pro klasifikaci zbytečné.
- **Transformace dat** – neboli převod dat. Zpravidla jde o zobecnění dat, například převod čísla na diskrétní hodnoty (interval). Například příjem z přesné výše na nízký/vysoký. Speciálním případem transformace dat je tzv. normalizace, kdy se obecný interval převádí na interval <0,1>, případně na interval <-1,1>.

3.1.2 Srovnávání klasifikačních metod

Ke srovnávání různých klasifikačních metod používáme převážně následující kriteria:

- **Přesnost předpovědi** – v kolika procentech daný model správně klasifikuje data, která nebyla v trénovací množině (nová data).
- **Rychlost** – výpočetní složitost pro vygenerování a následné používání klasifikačních pravidel.
- **Robustnost** – schopnost odolávat chybějícím datům a datovému šumu.
- **Stabilita** – schopnost vytvořit správný model pro velký objem dat.
- **Interpretovatelnost** – složitost daného problému pro jeho pochopení.

3.2 Predikce

Predikce je proces, který umožní přiřazovat datům hodnoty, které mají obecně spojitý charakter. Příkladem predikce je například určení výše platu daného pracovníka na základě znalostí dalších jeho vlastností. Nejznámější metodou pro predikci je tzv. lineární jednoduchá a lineární násobná regrese. Spousta nelineárních problémů jde často na tyto typy regrese transformovat [Zendulka].

Oba tyto typy dolovacích úloh tedy patří mezi tzv. prediktivní. Rozdíl mezi predikcí a klasifikací spočívá tedy v tom, že klasifikace se zabývá rozdělováním objektů do tříd na základě jejich vlastností, zatímco predikce předpovídá přesnou hodnotu zpravidla spojitého charakteru, například již zmiňovanou výši příjmu.

3.3 Klasifikační metody

Klasifikace může probíhat pomocí celé řady metod. Mezi nejznámější patří klasifikace pomocí rozhodovacího stromu, naivní bayesovské klasifikace, klasifikace pomocí neuronových sítí, klasifikátory založené na k-nejbližším sousedství, genetické klasifikátory a řada dalších.

Jako součást tohoto diplomového projektu byla zvolena implementace modulu pro naivní bayesovskou klasifikaci. Naivní bayesovské klasifikaci se budeme podrobněji věnovat v následující kapitole.

3.4 Naivní bayesovská klasifikace

Tato metoda provádí klasifikaci na principu, který je založen na *statistice*. Pracuje tak, že pro každý nový prvek, který není zařazen do žádné třídy, bude statisticky určeno to, s jakou pravděpodobností patří do jednotlivých tříd, do kterých klasifikujeme. Prvek je poté zařazen do třídy, pro níž je nejvyšší pravděpodobnost, že do ní daný prvek náleží. V celém tomto procesu se využívá tzv. *podmíněné pravděpodobnosti*.

3.4.1 Podmíněná pravděpodobnost a bayesův vzorec

Nechť X a Y jsou dva obecně různé jevy. Symbolem P(X|Y) budeme značit podmíněnou pravděpodobnost jevu X za podmínky výskytu jevu Y. Tato hodnota udává, jaká je pravděpodobnost, že nastane jev X, pokud bezpečně víme, že nastal jev Y. Podmíněná pravděpodobnost P(X|Y) se vypočítá pomocí vzorce:

$$P(X \mid Y) = \frac{P(X \cap Y)}{P(Y)}, \tag{1}$$

kde P(X ∩ Y) je pravděpodobnost, že nastanou jevy X a Y současně; P(Y) je pravděpodobnost jevu Y[Zendulka].

Je nutné si také uvědomit, že obecně hodnota P(A|B) není shodná s hodnotou P(B|A). Vyjádřením výrazu P(X ∩ Y) ze vzorců $P(X \mid Y) = \frac{P(X \cap Y)}{P(Y)}$ a $P(Y \mid X) = \frac{P(Y \cap X)}{P(X)}$ a následným jejich porovnáním dostaneme po úpravě vztah:

$$P(X \mid Y) = \frac{P(Y \mid X)P(X)}{P(Y)}. \tag{2}$$

Vztah (2) nazýváme bayesovským vzorcem. Tento vzorec hraje velkou roli při zařazování objektů do tříd při použití bayesovské klasifikaci.

3.4.2 Jednoduchá bayesovská klasifikace

Nyní si popíšeme princip a způsob, jakým může probíhat klasifikace na základě jednoduché bayesovské klasifikaci. Jednoduchou bayesovskou klasifikaci můžeme též nazývat naivní bayesovskou klasifikací.

Zaveďme si následující značení:

S	značí množinu všech vzorků (trénovacích dat).
$C_1, C_2, \ldots, C_m$	značí jednotlivé třídy, do kterých jsou vzory z množiny S klasifikovány. Symbol m tedy udává celkový počet těchto tříd.
$A_1, A_2, \ldots, A_n$	značí jednotlivé atributy, pomocí kterých bude prováděna klasifikace.
$X = (x_1, x_2, \ldots, x_n)$	kde x_i je hodnota atributu A_i pro všechna i = 1,…, m, značí testovací vzorek, který má být klasikovým do nějaké třídy.
s_i	značí počet prvků z množiny S, který je klasifikován do třídy C_i, kde i = 1,…, m; s_1 tedy například značí počet prvků z množiny S, který je klasifikován do třídy C_1.

Potom $P(C_i|X)$ pro libovolné $i = 1,\ldots, m$ udává podmíněnou pravděpodobnost toho, že libovolný vzorek padne do třídy C_i víme-li, že vzorek má atributy shodné s prvkem X. Neboli, jaká je pravděpodobnost, že prvek X patří do třídy C_i Úkolem je tedy najít takové i, pro které je hodnota $P(C_i|X)$ největší, protože je největší pravděpodobnost, že prvek X patří právě do této třídy. Prvek potom bude klasifikován právě do této třídy C_i s největší hodnotou $P(C_i|X)$ [Zendulka].

Podle Bayesova vzorce tedy platí:

$$P(C_i \mid Y) = \frac{P(X \mid C_i)P(C_i)}{P(X)}. \tag{3}$$

Pro určité číslo i je tedy výraz $P(C_i|X)$ maximální právě tehdy, když je maximální výraz $P(X|C_i)P(C_i)$, protože P(X) je konstantní pro daný vzorek X.

Je ovšem nutné rozlišovat $P(C_i|X)$ a $P(X|C_i)$. $P(C_i|X)$ nám udává, jaká je pravděpodobnost, že prvek X náleží do třídy C_i, kdežto $P(X|C_i)$ nám udává, jaká je pravděpodobnost, že libovolný prvek vybraný ze třídy C_i bude mít shodné hodnoty atributů jako prvek X.

Hledáme tedy třídu C_i, pro kterou je výraz $P(X|C_i)P(C_i)$ maximální.

- $P(C_i)$ je pravděpodobnost, že libovolně zvolený prvek patří do třídy C_i. Ta se tedy dá jednoduše určit pomocí vzorce: $P(C_i) = si / |S|$, kde $|S|$ značí kardinalitu množiny S, neboli počet prvků, který tato množina obsahuje.

- $P(X|C_i)$ je pravděpodobnost, že libovolný prvek vybraný ze třídy C_i bude mít shodné hodnoty atributů jako prvek X. Tuto pravděpodobnost získáme tedy jako součin pravděpodobností $P(x_k|C_i)$ pro k = 1,…, n, kde $P(x_k|C_i)$ je pravděpodobnost, že prvek, jehož atribut A_k je roven hodnotě x_k, bude zařazen do třídy C_i. Což formálně zapisujeme jako:

$$P(X \mid C_i) = \prod_{k-1}^{n} P(\mathrm{x}_k \mid C_i) \tag{4}$$

Hodnoty $P(x_k|C_i)$ následně určujeme podle typu atributu A_k, mohou nastat dva případy:

1. **Atribut A_k má diskrétní charakter**
   V tomto případě se $P(x_k|C_i) = s_{ik}/s_i$, kde s_{ik} je počet vzorků z množiny S zařazené do C_i, jejichž atribut A_k má hodnotu x_k.

2. **Atribut A_k má spojitý charakter**

$$\mathrm{P}(\mathrm{x}_k \mid \mathrm{C}_i) = \mathrm{g}(\mathrm{x}_k, \mu_c, \sigma_{c_i}) = \frac{1}{\sqrt{2\pi\sigma_{c_i}}} e^{=\frac{(x_k=\mu_c)^2}{2\sigma^2}}, \tag{5}$$

kde $g(x_k, \mu_c, \sigma_{c_i})$ je Gaussovo normální rozložení pro atribut A_k s hodnotu x_k. μ_c je průměrná hodnota a σ_{c_i} směrodatná odchylka pro atributy A_k patříci do třídy C_i [Zendulka].

3.4.3 Laplaceova korekce

Když se podíváme na vzorec (4)

$$P(X \mid C_i) = \prod_{k-1}^{n} P(\mathrm{x}_k \mid C_i),$$

Tak zjistíme, že může vyvstat jeden závažný problém. A to konkrétně v případě, že pro nějaké k platí: $P(x_k \mid C_i) = 0$, potom $P(X \mid C_i) = 0$.

Pro řešení tohoto problému používáme tzv. *laplaceovu korekci*. Laplaceova korekce není nic jiného než přidání jednoho prvku do každé množiny. Například máme-li množinu 1000 prvků s atributem příjem, kde 0 prvků příjem = „malý“, 990 prvků příjem = „střední“ a 10 prvků příjem = „vysoký“, potom *P*(příjem = „malý“) = 1/1003, *P*(příjem = „střední“) = 991/1003 a *P*(příjem = „vysoký“) = 11/1003.

Celkově tedy můžeme bayesovskou klasifikaci shrnout jako metodu vhodnou pro implementaci, jejíž výhodou je, že v řadě případů dává velice dobré výsledky. Jako nevýhodu je ovšem nutno zmínit, že tento způsob klasifikace předpokládá, že dané atributy, které používáme, na sobě nejsou závislé. V praxi ovšem někdy atributy závislé jsou, pro tyto případy tedy nebude tento způsob klasifikace vhodný.

4 Použité technologie

V této kapitole jsou popsány důležité technologie, které byly použity při implementaci tohoto diplomového projektu.

4.1 Oracle Data Mining

Celá aplikace *Dataminer* je aplikací s architekturou klient – server. Jako server je použit *Oracle server*. *Oracle Data Mining* je součástí relačního databázového systému od firmy Oracle. Obsahuje podporu pro dolování z dat, které jsou uloženy v databázi tohoto systému [Mader].

Dolování z dat je v tomto případě plně v kompetenci serveru dříve nazývaného *Data Mining server* je od verze 10.2 nazýván *Data Mining Engine* (dále jen. DME). Klient pouze uloží data do databáze na serveru. Poté vytvoří popis dolovací úlohy jako takové a odešle jej na server (DME). V tomto okamžiku je veškerý výpočet prováděn na straně DME a klient jen po dokončení procesu dolování požádá server o výsledek. Komunikace se serverem může probíhat dvěma způsoby. První z nich je komunikace pomocí jazyka PL/SQL a druhý, který se používá i v tomto projektu, je komunikace pomocí knihoven jazyka *Java*. Použití jazyka Java plyne z toho, že celý projekt je vystavěn na platformě *NetBeans* a implementován je v jazyce Java.

Od verze Oracle 10.2 je nejen pro tuto komunikaci, ale i vývoj dataminingových aplikací implementován standard *Java Data Mining* označovaný JSR 73. Java Data Mining určuje, jak má vypadat rozhraní dolovacího systému, aby byla zaručena nezávislost aplikace na daném nástroji pro datamining. Tento standard má JavaDoc dokumentaci, je k němu vytvořena i sada demonstračních příkladků, které napomáhají uživateli pochopit, jak co nejlépe a nejefektivněji využít dolovacího systému ODM.

Oracle Data Miner od verze 10gR2 umožňuje použití širokého spektra dolovacích funkcí. Také obsahuje řadu různých operací, jako je transformace, vzorkování a binning dat. Dále algoritmy pro prediktivní i deskriptivní úlohy viz Tabulka 4.1.1 a 4.1.2. a také algoritmy pro textové a prostorové dolování a specializované úlohy, například porovnání proteinových řetězců a sekvencí DNA.

Prediktivní úlohy	
Klasifikace	Rozhodovací strom, naivní bayesovská klasifikace, bayesovské sítě a SVM (support vector machines)
Regrese	SVM
Detekce odlehlých hodnot	One-class SVM

Tabulka 4.1.1 *Prediktivní metody podporované ODM*

U všech prediktivních úloh je možnost zobrazit výsledky několika různými způsoby. Je možné použít různé druhy *liftů* (kvantilový, komutativní, nekomutativní aj.). Další možností je tzv. *confusion matrix*, ve které je uvedeno, v kolika případech byla daná metoda úspěšná. Další možností je využití tzv. *ROC* (*Receiver operating characteristic*) k porovnání více klasifikačních metod mezi sebou.

Deskriptivní úlohy	
Asociační analýza	Algoritmus Apriori
Shlukování	K-means, Orthogonal Partitioning Clustering

Tabulka 4.1.2 *Deskriptivní metody podporované ODM*

4.2 Platforma NetBeans

NetBeans je úspěšným open source projektem, který založila firma *Sun Microsystems* v červnu 2000. V současné době existují dva produkty. Prvním z nich je vývojové *prostředí* NetBeans (NetBeans IDE) a druhým je *vývojová platforma* NetBeans (The NetBeans Platform) [NetBeans].

Vývojové prostředí NetBeans IDE je nástroj umožňující vývojářům psát, překládat a ladit a distribuovat aplikace. Samotné vývojové prostředí je vyvíjeno v jazyce *Java*, ale podporuje v podstatě jakýkoli programovací jazyk jako například *C*, *C++*, *PHP*, *JavaScript*, *Ruby*. Samozřejmě také jazyk Java a všechny jeho platformy, tedy *J2SE*, *J2EE* i *J2ME*. Toto vývojové prostředí je také velmi vhodné k tvorbě GUI aplikací využívající knihovny *Swing* a *AWT*.

Vývojová platforma NetBeans je modulární a rozšiřitelný základ pro tvorbu rozsáhlých desktopových aplikací. NetBeans platforma stejně jako vývojové prostředí NetBeans IDE je šířena pod licencí Open Source a je možné je využívat v komerčním i nekomerčním prostředí zcela bezplatně. Zdrojové kódy jsou dostupné pod licencemi *Common Development and Distribution License* (*CDDL*) v1.0 a *GNU General Public License* (*GPL*) v2.

Platforma poskytuje vývojářům celou řadu služeb desktopových aplikací a umožňuje tím, že se vývojář může plně věnovat podstatným funkcím aplikace. Platforma má následující základní vlastnosti [Mader]:

- Správa uživatelského rozhraní
- Správa uživatelského nastavení
- Správa práce s okny
- Podpora ukládání
- Práce s tzv. průvodci (wizards)

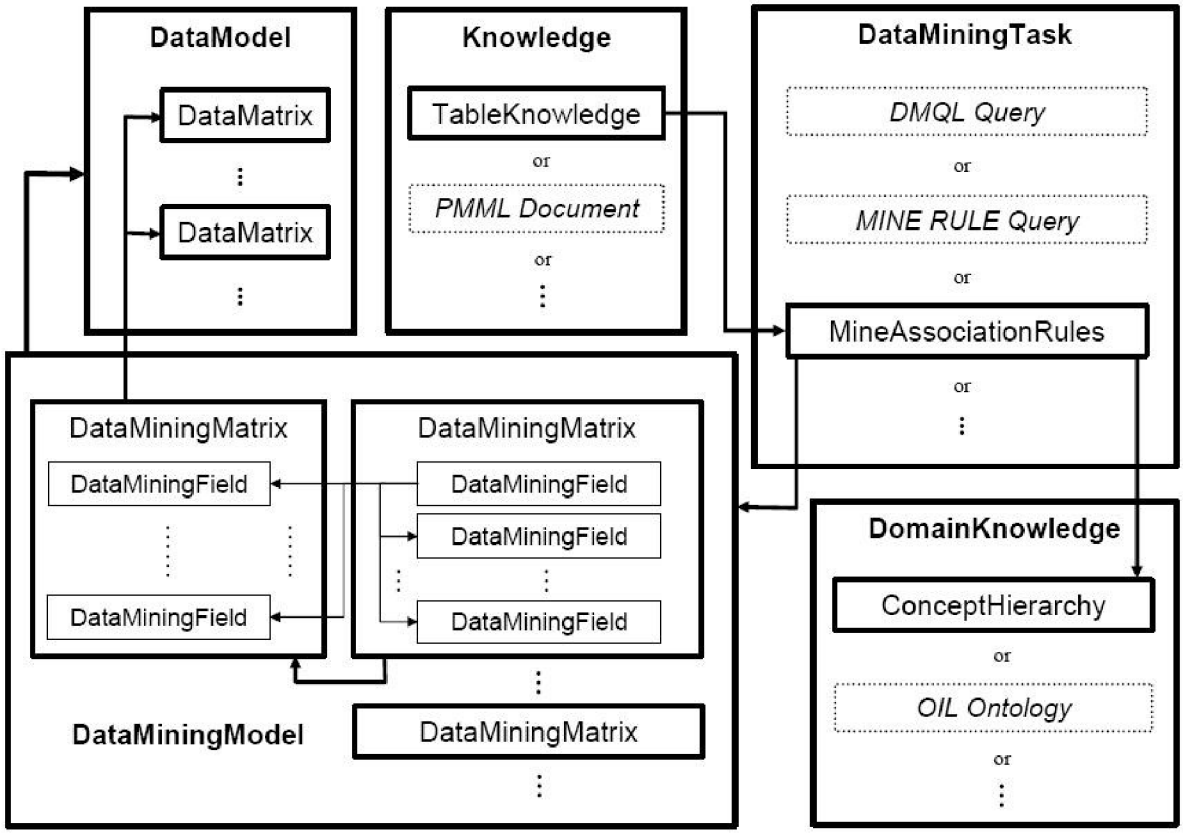
4.3 Jazyk DMSL

Jazyk *DMSL* (z angl. *Data Mining Specification Language*) je jazyk založený na *XML* a slouží k uložení nastavení a popisu celého dolovacího procesu. Umožňuje nám definovat vstupní data, popisovat transformace, znalosti o vstupních datech tzv, *domain knowledge*, dolovací úlohu a získané znalosti – podrobnější informace o DMSL jsou uvedeny v práci pana Kotáska [Kotásek]. Pro každou část má jazyk DMSL odpovídající element. Navíc je přidán jeden element pro definici funkcí, které budou v dolovacím procesu použity. Rozšiřitelnost a úprava je díky použití jazyka XML velice jednoduchá. Vzájemné vztahy mezi všemi elementy jsou znázorněny na obrázku 4.3.1.

Nyní si popíšeme blíže jednotlivé elementy.

DataModel – tento element slouží k popisu vstupních dat pro daný dolovací proces. Tento element může obsahovat další element tzv. *datových matic* (*DataMatrix*), který reprezentuje výběr dat v rámci jedné databázové tabulky. Každý vybraný sloupec tabulky je reprezentován elementem *DataField*, který popisuje jeho vlastnosti. Může se odkazovat na element *FunctionPool*.

DataMiningModel – tento element popisuje ta data, která budou pro samotný proces dolovaní použita, tedy i odvozené sloupce a transformace dat z *DataField*. *DataMiningModel* se vždy odkazuje na právě jeden *DataModel*, který je pro něj reprezentuje zdroj vstupních dat. Dále se také může odkazovat na jeden, nebo žádný *FunctionPool*. *DataMiningModel* obsahuje element *DataMiningMatrix*, ten se skládá z *DataMiningFields*.



Obrázek 4.3.1 *Vztahy mezi elementy jazyka DMSL* – převzato z [Kotásek]

DomainKnowledge – obsahuje informace (znalosti) o datech jako jsou integritní omezení, jejich vztahy, obory hodnot a další. Tento element má volnou syntaxi a v současné době není v projektu nijak využíván.

Knowledge – také element s volnou syntaxí. Slouží k reprezentaci získané znalosti, jeho definice a syntaxe závisí na typu dolovací úlohy. Jediné povinné atributy jsou atributy *language* a *name*.

DataMiningTask – opět element s volnou syntaxí. Má také dva povinné atributy a těmi jsou použitý jazyk a jméno.

FunctionPool – atribut specifikující funkce, které mohou být aplikovány na vstupní data při jejich čištění a předzpracování. Každý *FunctionPool* má své jednoznačné jméno, pomocí kterého se na něj mohou odkazovat ostatní elementy. Mimo funkcí z *FunctionPoolu* lze využít i tzv. inline funkcí definovaných přímo v místě použití.

V této části jsme se dozvěděli pouze základní informace o jazyku DMSL a jeho použití. Pro podrobnější informace a nastudování dané problematiky je vhodné nastudovat práci pana Kotáska [Kotásek].

5 Systém vyvíjený na FIT

V této části se budeme věnovat systému pro dolování z dat, který je v současnosti vyvíjen na Fakultě informačních technologií VUT v Brně. Tento systém je již několik let vyvíjen v rámci semestrálních a diplomových prací.

Jako první vznikl systém pro dolování z dat, který pracoval s *MySQL* databází. To ovšem narazilo na problém, že databázový systém MySQL nemá žádnou podporu pro dolování z dat a vše muselo být řešeno přes pomocné tabulky, což se jevilo jako nevhodné. V dalším vývoji se tedy od MySQL upustilo a bylo navrženo jádro systému, které pracuje s databázovým systémem firmy Oracle. Databázový systém Oracle umožňuje použití modulu Oracle Data Mining, který poskytuje možnost využití jeho podpory dolování z dat na jeho databázovém serveru. Postupně se směřovalo k využití databáze Oracle verze 10.2. Dále je v tomto projektu využito jazyka DMSL

V dalších semestrálních projektech a pracích prošel vývojem a řadou změn. Implementace systému spolupracujícího s Oracle databází byla započata Ing. Doležalem [Doležal]. Jeho práce byla zaměřena hlavně na vývoj jádra celého systému. Systém byl formulářová aplikace, která umožňovala základní kroky dolovacího procesu. Implementován byl výběr dat z databáze a jejich předzpracování. Byly položeny i teoretické základy práce s dolovacími moduly. Tato část byla opravdu teoretická a tehdy vytvořený modul byl pouze triviální.

Na tuto práci navázal Ing. Gálet [Gálet], kdy nad jádrem vytvořil aplikaci s grafickým uživatelským rozhraním na platformě NetBeans. Jednou z hlavních změn bylo zavedení reprezentace jednotlivých kroků dolovacího procesu pomocí komponent grafu dolovacího procesu. K tomu bylo i zapotřebí pozměnit DMSL tak, aby bylo možno ukládat i informace o grafické reprezentaci dolovacího procesu.

Jako poslední na tomto projektu pracovali Ing. Krásný[Krásný] a Ing. Mader[Mader]. Ing. Krásný se zabýval úpravou jádra stávajícího systému a systému jako celku tak, aby se zvýšila jeho použitelnost. Implementoval celou řadu změn a došlo k rozsáhle přestavbě jádra tak, aby byla umožněna rozšiřitelnost a udržovatelnost systému. Pro tuto diplomovou práci je nejpodstatnější změnou to, že dolovací modul již není externí třídou, o kterou se stará jádro, ale stal se rovnocennou komponentou grafu celého dolovacího procesu. Ing. Mader se zabýval vývojem dolovacího modulu pro asociační pravidla. Vytvořil koncepci toho, jak vhodně připojit dolovací modul k systému a napsal podrobnou dokumentaci tohoto problému.

Celý systém se dá rozdělit do několika částí, nyní si blíže popíšeme každou z nich.

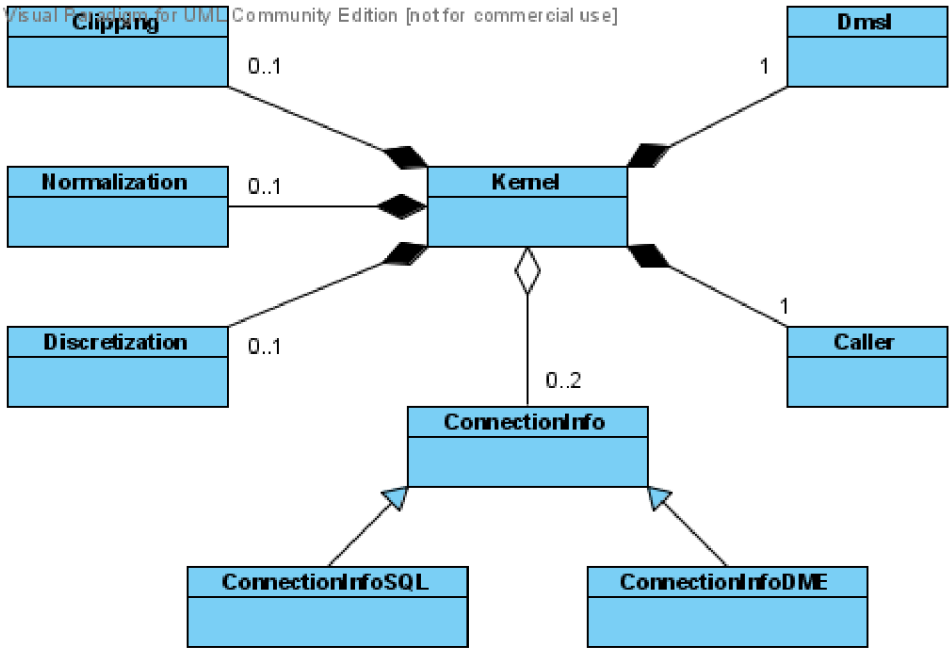
5.1 Jádro systému

Třída jádra *kernel.java* prošla díky práci Ing. Krásného řadou změn. Úkolem této třídy je poskytovat připojení k serveru – jedno pro SQL dotazy a jedno pro DME transformace [Krásný]. Jádro dále poskytuje vazbu na třídy *ConnectionInfo*, která má zajišťovat připojení k serveru. Dále ke třídě *Caller* pro volání zkompilovaných interních funkcí a pro přístup k datové struktuře Také k instancím pro transformaci dat *Clipping*, *Normalization* a *Discretization*.

Jádro tedy poskytuje tyto funkce a umožňuje:

- Zadávat dolovací metody
- Načítání a ukládání do DMSL
- Organizovat práci do projektu
- Provádět transformace dat

- Přidávat dolovací moduly

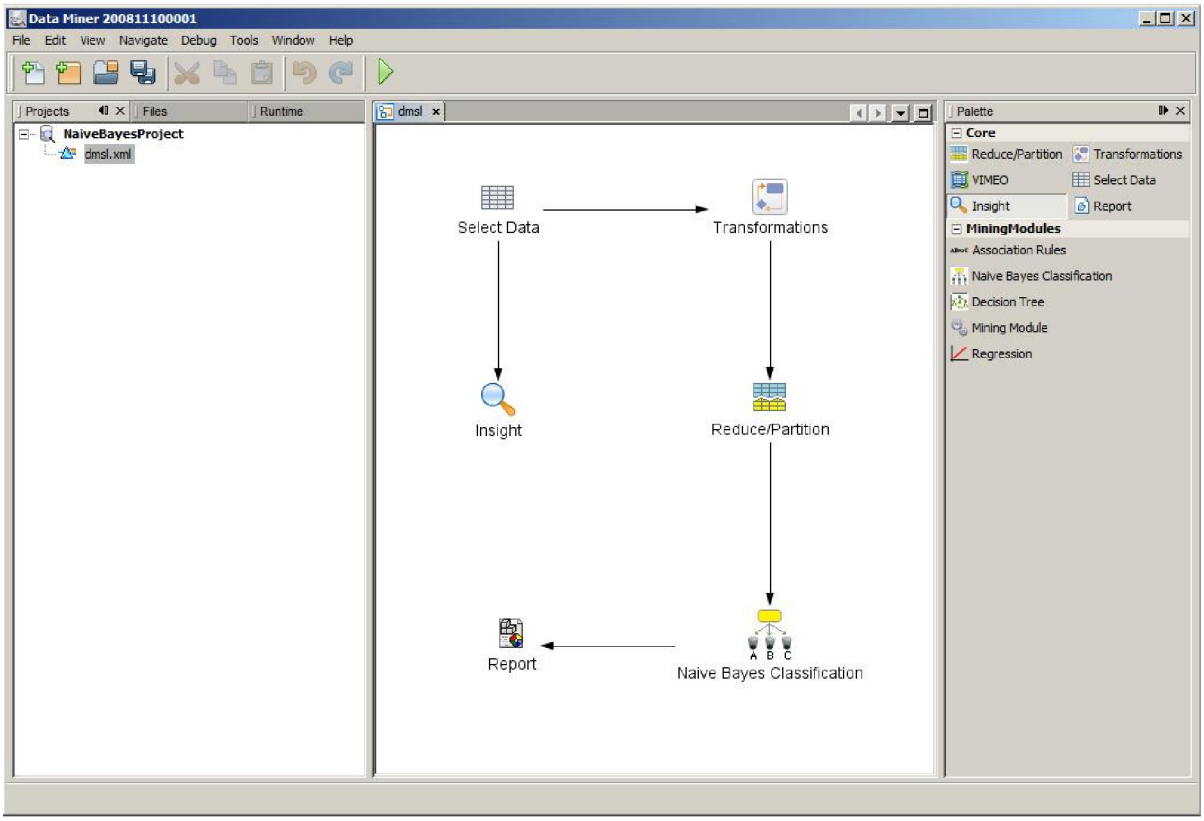


Obrázek 5.1.1 *Diagram tříd kernel.java* – převzato z [Krásný]

5.2 Grafické uživatelské rozhraní systému

Jádro bylo od počátku navrženo tak, aby k němu bylo možné připojit nezávislé GUI, které není jeho součástí a dalo se lehce nahradit jiným. Je postaveno na knihovnách Swing a AWT. Původní verze aplikace měla základní rozhraní, které sloužilo pouze k demonstraci funkčnosti implementovaného jádra. Byla to formulářová aplikace, kde se pomocí záložek dá přepínat mezi jednotlivými funkcemi. Tato koncepce je však pro potřeby aplikace nedostačující a řešení složitějších úloh by bylo značně komplikované. Celé rozhraní bylo proto nahrazeno rozhraním novým.

Tohoto úkolu se zhostil svým diplomovým projektem Ing. Gálet. Použil knihovnu *NetBeans Visual Library*, která je určena k tvorbě grafů a diagramů. Podařilo se mu vybudovat prostředí, které umožňuje vkládat komponenty znázorňující jednotlivé kroky dolování, ty mezi sebou propojovat a dal tím základ budoucí aplikaci, která má velký potenciál stát se kvalitním komplexním grafickým nástrojem pro dolování dat. Grafické uživatelské rozhraní by se dalo přirovnat ke grafickému uživatelskému rozhraní aplikace *SAS Enterprise Miner*. Zavedl organizaci do projektů, integroval do DMSL dokumentu grafická metadata ve formě poznámek a nasměroval projekt správným směrem. Byl však omezen přílišnou specifičností jádra a z pohledu dolování nebylo možné přinést další funkčnost [Krásný 2008]. Tento nedostatek byl odstraněn v rámci řešení diplomové práce Ing. Krásným.



Obrázek 5.2.1 *Ukázka GUI systému*

5.3 Dolovací moduly

Dolovací modul se po inovacích Ing. Krásného stal běžnou komponentou grafu, který znázorňuje danou dolovací úlohu. Připojení modulu jako prvku takového grafu probíhá za pomoci abstraktní třídy *MiningPiece*, která svým potomkům poskytuje všechny prostředky pro připojení do grafu. Dolovací modul je tedy běžnou komponentou a pro práci s ním není potřeba žádné zvláštní zacházení [Krásný].

Modul má tedy za úkol implementovat danou klasifikační metodu a být nezávislou komponentou. Musí ovšem být kompatibilní a spolupráce schopný s již existujícím systémem. Moduly pak provádí samotné dolování dat, při kterém využívají služeb Oracle serveru. Jeho implementace a integrace do systému je součástí této diplomové práce.

5.4 Další komponenty dolovacího procesu

V této kapitole zmíníme další komponenty dolovacího procesu. S jednou z nich jsou již zmíněné dolovací moduly. V současné době jsou implementovány dva dolovací moduly. První slouží pouze pro testovací účely a neimplementuje žádnou dolovací metodu. Druhý modul je již plně funkční a slouží pro dolování asociačních pravidel. Jeho implementace byla náplní diplomové práce ing. Madera [Mader].

Nyní si tedy představíme další komponenty, které mohou být součástí dolovacího procesu.

Select data

Tato komponenta umožňuje výběr vstupních dat. Po otevření této komponenty se v levé části zobrazí tabulky, které jsou vytvořeny v databázi, ke které jsme připojeni. Z těchto tabulek je možno vybrat

libovolné sloupce. Lze také využívat vytváření podmínek pro spojování tabulek v databází. Tato komponenta je nutná pro každý dolovací proces, protože dolování nemá bez vstupních dat žádný smysl.

Transformations
Jak již název napovídá, tato komponenta umožňuje transformovat vstupní data. Na tato data lze aplikovat řadu ODM transformací, jako je diskretizace, normalizace a clipping. Pomocí této komponenty je také možné definovat nové sloupce, stejně tak jako vynechávat některé sloupce ze vstupních dat.

Reduce/Partition
Tato komponenta umožňuje rozdělit vstupní data na data pro vytvoření modelu (Standard/Training data) a na data pro testovaní (Testing data) a to v uživatelem zadaném poměru. Tato komponenta je nutná pro všechny dolovací moduly, které obsahují fázi tvorby modelu a fázi jeho testování. Pomocí této komponenty je také možné snížit objem dat a to tak, že vybranou metodou odstraníme zadanou procentuální část vstupních dat.

Report
Tato komponenta slouží k prezentaci výsledků dolovacího procesu. Každý dolovací modul má specifické požadavky na zobrazení výsledku, které plynou z vlastností daného dolovacího algoritmu. Report je tedy komponenta, která zajistí otevření okna s výsledky zvoleného dolovacího modulu (algoritmu) a její obsah je pro každý dolovací modul jiný.

Insight
Tato komponenta umožňuje zobrazovat data z tabulek, které jsou fyzicky vytvořeny v databázi komponentami pro výběr a předzpracování vstupních dat atp.

Vimeo
Tato komponenta umožňuje přidávat k jednotlivým sloupcům VIMEO funkce. Umožňuje přístup k těmto funkcím a jejich editace. Komponenta funguje jako datový filtr.

6 Naivní bayesovská klasifikace a Oracle Data Mining

Obsahem této práce je vytvoření dolovacího modulu pro naivní bayesovskou klasifikaci, neboli modulu používající jeden z algoritmů s podporou, kterou poskytuje přímo Oracle Data Mining. K implementaci tohoto modulu je nutné podrobné nastudování problematiky, týkajících se naivní bayesovské klasifikace jako takové a dále potom také konkrétní možnosti podpory pro datamining od Oracle Data Mining, konkrétně potom možností využití *Java API*. K tomu lze s výhodou využít vzorových příkladu, které Oracle poskytuje ve své instalaci. Tyto příklady obsahují ve zjednodušené podobě jednotlivé algoritmy, které Oracle Data Mining podporuje. V příkladech je podrobně vysvětlen celý dolovací proces v jednotlivých krocích. Od přípravy vstupních dat pro daný algoritmus, přes samotný dolovací proces až po ukázkovou textovou prezentaci výsledků dolování. Pro všechny tyto příklady je nutné mít v databázi vytvořené tzv. *Sales History schéma* (SH schéma), které je součásti instalace Oracle Data Mining.

Funkce	Algoritmus
Asociační pravidla	Apriory
Dolování v textu	Non-Negative Matrix Factorization
	Support Vector Machines
Důležitost atributu	Minimum Description Lenght
Feature Extraction	Non-Negative Matrix Factorization
Klasifikace	Adaptivní Bayesovská síť
	Naivní Bayesovská klasifikace
	Rozhodovací strom
	Support Vector Machines
Regrese	Support Vector Machines
Shlukování	K-Means
	O-Cluster

Tabulka 6.1 *Přehled ukázkových příkladů ODM*

Další významnou pomůckou k pochopení dané problematiky je i rozšiřující aplikace Oracle Data Miner. Jedná se o aplikaci vyvinutou firmou Oracle, která umožňuje zákazníkům analyzovat svá data [Oracle a], stejně tak ale umožňuje nastudovaní algoritmů, které Oracle Data Mining poskytuje a to v podobě přehledných průvodců. Zmínění průvodci uživatele provedou celým procesem, od výběru vstupních dat, přes nastavení parametrů daných algoritmů až po prezentaci získaných výsledků. Díky tomu zjistíme, jaké parametry daný algoritmus potřebuje mít zadány, dále jaký je formát vstupních dat a jaké mohou být výstupy dolovacího procesu.

Dále lze s výhodou použít nástroj *Oracle SQL Developer*. Jedná se o grafický nástroj pro správu databází, který je zdarma. SQL developer umožňuje prohlížet databázové projekty, spouštět SQL příkazy a skripty[Oracle b]. Tento nástroj zvyšuje efektivitu a zjednodušuje práci. Spolu se SQL

developerem je vhodné používat konzolu pro správu databáze *SQL*Plus*. SQL*Plus je součástí aplikace *InstantClient* a na rozdíl od SQL developeru podporuje více datových typů, jako jsou například typ kategorie *Nested*, které SQL developer není schopný vypsat. Pro rychlou a efektivní práci je vhodné společné použití těchto dvou nástrojů a jejich vzájemné propojení. SQL developer umožňuje nastavit cestu k externí SQL*Plus konzoli, díky čemuž se poté konzole připojuje ke stejné databázi, ke které je právě přihlášen SQL Developer.

6.1 Naivní bayesovská klasifikace

Teoretickými základy k naivní bayesovské klasifikaci jsme se zabývali v kapitole 3.4. Nyní si její princip ukážeme na reálném příkladě a zmíníme některé významné parametry naivní bayesovské klasifikace pro ODM.

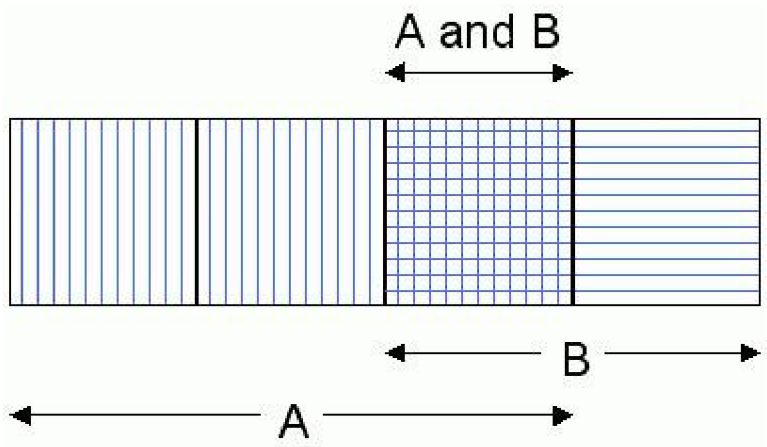
Bayesovská klasifikace využívá tzv. bayesova vzorce (2) v kapitole 3.4.1. Princip si předvedeme na příkladu navýšení výdajů [Oracle c].

Příklad 6.1.1: Představme si, že potřebujeme zjistit pravděpodobnost, že zákazník mladší 21 let navýší své výdaje. V takovém případě máme tedy dva jevy. První jev (A) je, že je zákazník mladší 21 let a druhý jev (B) je, že zákazník navýší útratu.

- Jestliže máme v testovacích datech 100 zákazníků a 25 z nich má méně než 21 roků a navýší svou útratu, potom $P(A \cap B) = 0.25$.
- Dále pak, jestliže v testovacích datech máme mezi zmiňovanými 100 zákazníky 75 zákazníků mladších 21 let, potom P(A) = 0.75

Z těchto pravděpodobností můžeme pomocí bayesova vzorce vypočítat pravděpodobnost P(B|A) = 25/75 = 1/3.

Obecně si můžeme tento postup znázornit pomocí jednoduchého obrázku.



Obrázek 6.1.1 *Znázornění pravděpodobností jevů* – Přepracováno z [Oracle c]

Jednotlivé pravděpodobnosti jevů z obrázku 6.1.1 jsou potom tedy:

P(A) = 3/4 = 0.75

P(B) = 2/4 = 0.5

$P(A \cap B)$ = 1/4 = 0.25

P(A|B) = $P(A \cap B)$ / P(B) = (1/4) / (2/4) = 0,5

P(B|A) = $P(A \cap B)$ / P(A) = (1/4) / (3/4) = $0.\overline{3}$

Nyní se dostáváme k již zmiňovaným parametrům. První z nich je tzv. *pairwise treshold*. Pojmem „pairwise“ jsou označovány ty případy, kdy se obě podmínky vyskytují společně. V příkladě 6.1.1 je 25% všech případů „pairwise“, neboli podmínka je mladší 21 let se vyskytuje společně s podmínkou, že zákazník navýší útratu. *Pairwise treshold* je poté minimální procentuální podíl výskytů pairwise pro vytvoření modelu.

Dalším parametrem je tzv. *singleton treshold*. Singleton je naopak od „pairwise“ označení pro případy, kdy se podmínka vyskytuje osamoceně. V příkladu 6.1.1 je „singleton“ 75% všech případů. Analogicky pak „singleton treshold“ je minimální procentuální podíl výskytů „singletonu“ pro vytvoření modelu.

Posledním parametrem, který v této části zmíníme jsou, tzv. *prior probabilities*. Někdy nejsou data, která máme k dispozici, zcela reprezentativní. Například můžeme mít v našem vzorku pouze pár uživatelů mladších 21 let, přitom ve skutečnosti víme, že zákazníků mladších 21 let existuje více, než znázorňuje testovací vzorek dat. Tento nedostatek můžeme právě kompenzovat specifikováním „prior probabilities“ ve fázi trénování modelu v dolovacím procesu. Neboli můžeme sami nastavit procentuální zastoupení v jednotlivých koších, do kterých jsou data rozdělena. S rozdělováním dat do košů se seznámíme v následující kapitole.

6.2 Příprava dat pro naivní bayesovskou klasifikaci

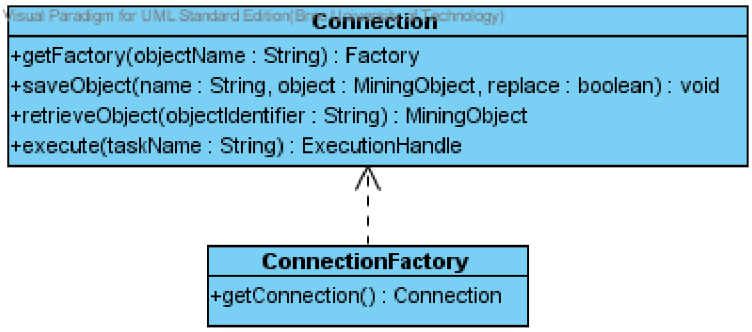
Je nutné si uvědomit, že naivní bayesovská klasifikace vyžaduje tzv. *binning*, neboli rozdělení dat do košů [Oracle c]. Naivní bayesovská klasifikace používá techniky na vyčíslení pravděpodobností, s jakými bude daný prvek do dané třídy patřit. Dále je nutné brát v potaz, že atributy by měly být rozděleny do vhodného počtu košů. Numerická data lze rozdělit do košů podle rozsahu hodnot. Například numerický atribut plat je možno rozdělit do tří košů: nízký, střední a vysoký. Stejně tak kategorické lze rozdělit do menšího počtu košů. Jako příklad lze uvést kraje namísto měst. Rozdělování do košů pomocí tzv. Equi-width se nedoporučuje, protože může zapříčinit, že se data rozdělí do pár košů, někdy dokonce do koše jediného, což značně snižuje sílu tohoto algoritmu.

Z toho plyne, že před spuštěním dolovacího modulu je třeba zajistit, aby do něj vstupovala data v této podobě. Je nutné také zvolit správný počet košů, jedině tak zajistíme, že získané výsledky, budou mít pro nás význam a budou dávat smysl.

6.3 Rozhraní ODM pro naivní bayesovskou klasifikaci

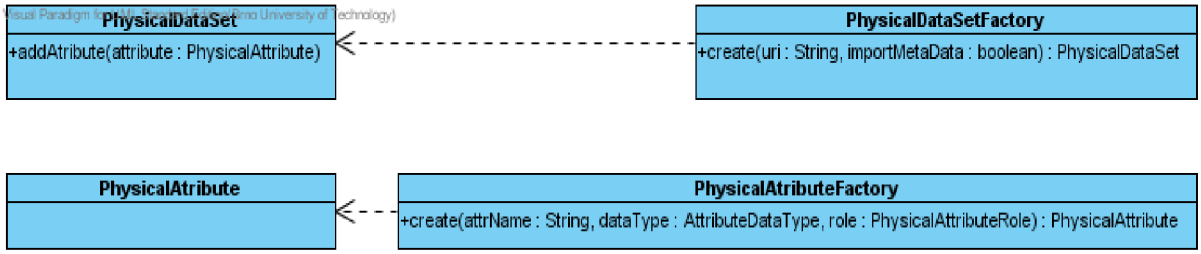
V této kapitole si blíže popíšeme stěžejní třídy, které jsou používány v rámci dolovacího algoritmu a jsou tedy nezbytné pro naivní bayesovskou klasifikaci. Nejprve si popíšeme, jak celý dolovací proces probíhá.

K úspěšnému dolování je nezbytné připojení k serveru a tím pádem i k databázi, která se na něm nachází a poskytuje potřebná data. O toto připojení se stará třída *Connection* z balíčku javax.datamining.resource, která je znázorněna na obrázku 6.3.1. Toto rozhraní umožňuje ukládání objektů do databáze, stejně tak umožňuje přístup k nim.



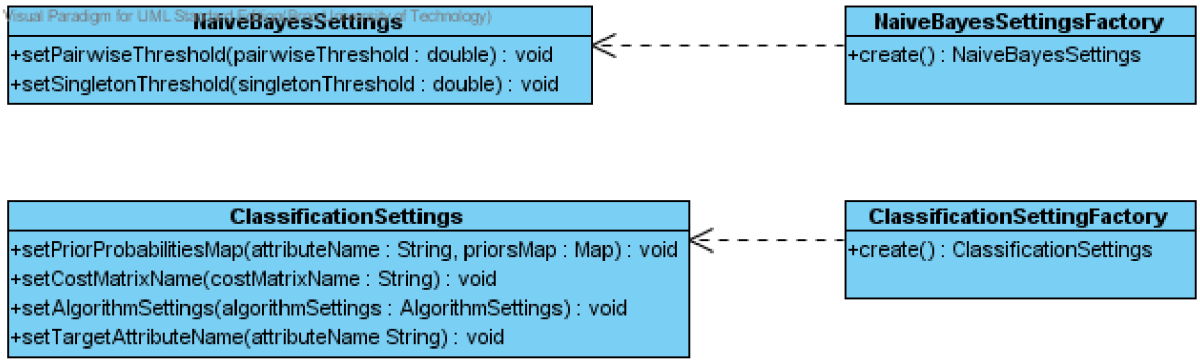
Obrázek 6.3.1 *Znázornění třídy Connection*

Poté co je nastaveno připojení k databázi se vytvoří objekt *PhysicalDataSet*. Rozhraní *PhysicalDataSet* slouží pro popis dat, které mají být použity jako vstup pro dolování z dat. Atributy fyzických dat jsou namapovány na logické atributy logického modelu dat. Data, které poskytuje *PhysicalDataSet* mohou být požita jak pro tvorbu modelu, tak i to pro testování a aplikaci.



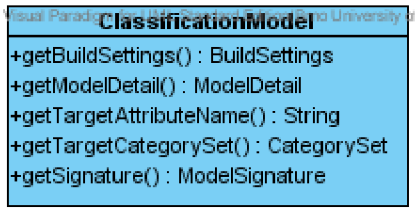
Obrázek 6.3.2 *Znázornění tříd PhysicalDataSet a PhysicalAtribute*

Po vytvoření *PhysicalDataSetu* je nutné nastavit nezbytné parametry pro naivní bayesovskou klasifikaci. O to se starají třídy *NaiveBayesSettings* a *ClassificationSettings*. Jak již názvy napovídají, třída *NaiveBayesSettings* se stará o nastavení parametrů specifických pouze pro naivní bayesovskou klasifikaci, jako jsou například *singleton treshold* a *pairwise treshold*, které jsme již zmiňovali výše. Zatímco třída *ClassificationSettings* se stará o nastavení parametrů, které používají všechny klasifikační algoritmy. Mezi ně patří nastavení cílového atributu, dolovacího algoritmu, pravděpodobností výskytu a dalších.



Obrázek 6.3.3 *Třídy NaiveBayesSettings a ClassificationSettings*

V dalším kroku se vytvoří objekt *ClassificationModel*, který obsahuje metada, která plynou z vytvoření modelu za pomocí *ClassificationSettings*. Neboli tímto vytvoříme samotný model pro naivní bayesovskou klasifikaci.



Obrázek 6.3.4 *Třída ClassificationModel*

V dalším kroku tento model otestujeme a tím získáme potřebné informace o jeho vlastnostech a jeho důležité metriky. K tomuto účelu slouží třída *ClassificationTestTask*. Ta se používá k testování vytvořeného modelu a zjišťuje jeho správnost. Mezi výsledky, které toto testování dodává, patří přesnost, tzv. *confusion matrix*, která udává poměry špatně a správně klasifikovaných prvku do jednotlivých tříd a další statistiky získané během testování.

Posledním krokem může být aplikace modulu na námi zadaná data. K tomu slouží objekt *ClassificationApplySettings*, který umožňuje nastavit požadované parametry pro aplikaci modelu na konkrétní data.

Celkově se tedy dá říci, že výstupem procesu naivní bayesovské klasifikace je model a jeho metriky. Tento model lze poté aplikovat na konkrétní data, která jsou zadána uživatelem. Na obrázcích v této kapitole. jsou zobrazeny stěžejní třídy, které jsou použity během samotného dolování za pomocí algoritmu pro naivní bayesovskou klasifikaci. Pro přehlednost jsou u daných tříd zobrazeny pouze metody a atributy, které jsou v rámci tohoto projektu používány.

7 Vytvoření nového dolovacího modulu

Náplní této kapitoly je seznámení s postupem tvorby nového dolovacího modulu a jeho následná integrace do aplikace *Dataminer*. Jak již bylo zmíněno dříve, aplikace Dataminer je vyvíjena na platformě NetBeans. Pro vývoj takového modulu je třeba tzv. *Netbeans Develpoment* verze zmiňovaného prostředí. Pro vytvoření dolovacího modulu pro Dataminer vytvoříme v NetBeans nový projekt *Module* z *NetBeans Modules*. V našem případě máme k dispozici zdrojové kódy celé aplikace Dataminer a celou Module Suit spojující vše do jednoho celku. Můžeme tedy dále zvolit možnost Add to Model Suite. Kdybychom neměli k dispozici celou Model Suite, museli bychom zvolit druhou možnost samostatně vyvíjeného modulu – *Standalone Module*. Kdybychom vytvářeli samostatný modul (Standalone module) museli bychom také zvolit NetBeanst Platform stejné verze, jaká odpovídá verzi NetBeans platformy aplikace Dataminer. Tím zajistíme přístup ke všem třídám a knihovnám potřebným k vývoji nového modulu [Mader]. Dále už jen vyplníme *Code Name Base* a název modulu a vytváření nového modulu pomocí NetBeans je dokončeno.

Následně je už jen třeba ve složce nového modulu vybrat složku *Libraries* a v ní přidat závislosti modulu pomocí volby *Add Module Dependency*. Z nabídky je třeba vybrat všechny následující možnosti: *Dialogs API*, *dm-api*, *dm-core-wrapper*, *dm-jfreechart-wrapper*, *dm-jmath-wrapper*, *UI Utilities API* a *Utilities API*. Všechny tyto jsou nezbytné pro další vývoj modulu.

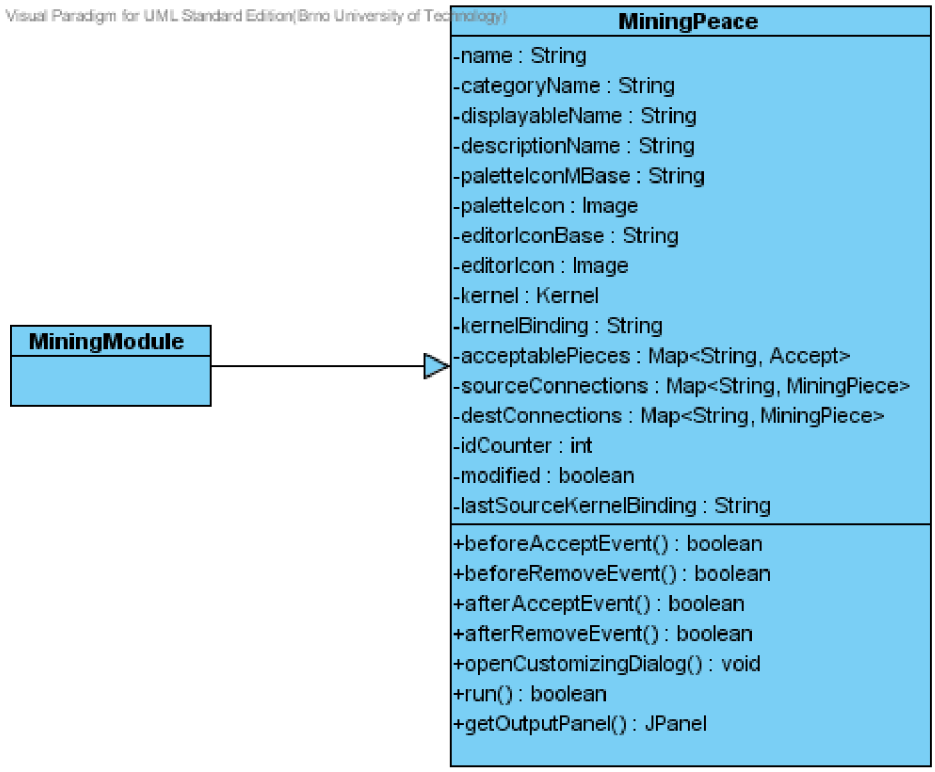
Nyní už můžeme takto nově vytvořený modul začlenit do aplikace dataminer a postupně začít implementovat jeho funkčnost. Předtím je ovšem nutné provést implementaci abstraktní třídy *MiningPeace* a upravit záznam v souboru *layer.xml*. Této problematice se budeme věnovat v následujících kapitolách.

7.1 Implementace abstraktní třídy MiningPeace

Chceme-li zpřístupnit modul i pro aplikaci Dataminer tak, aby jej bylo mono přidávat jako komponentu do grafu dolovacího procesu, je nutné nejprve implementovat nezbytné metody abstraktní třídy *MiningPeace*. Struktura této třídy je znázorněna na obrázku 7.1.1.

Každá třída projektu Dataminer, která dědí od třídy *MiningPeace*, představuje komponentu, kterou lze přidat do grafu znázorňujícího dolovací proces [Krásný]. V konstruktoru třídy, která dědí od *MiningPeace* je nutné nastavit následující:

- `setDisplayableName(NbBundle.getMessage(ClassifBCModule.class, "NAME_NazevModulu"));`
- `setDescription(NbBundle.getMessage(ClassifBCModule.class, "HINT_NazevModulu"));`
- `setEditorIcon("cz/vutbr/fit/dataminer/classifbcmodule/resources/Icon48.png");`
- `setPaletteIcon("cz/vutbr/fit/dataminer/classifbcmodule/resources/Icon16.png");`



Obrázek 7.1.1 *Znázornění třídy MiningPeace*

Prvním řádkem řekneme, že jméno modulu se nachází v lokalizačním balíku "*NAME_NazevModulu*". Stejně tak druhý bod říká, že nápověda je uložena v "*HINT_NazevModulu*". Poslední dva body udávají cestu k ikonám modulu. V třetím bodě je to cesta ke klasické ikoně, která bude zobrazena v grafu dolovacího procesu, která je uložena v souboru *Icon48.png*. Zatímco v posledním bodě je obsaženo nastavení cesty k miniatuře ikony, která se zobrazuje v paletě nástrojů aplikace Dataminer a je uložena v souboru Icon16. Konstruktor třídy tedy pouze nastaví název, nápovědu a ikony dané komponenty.

Aby ovšem aplikace Dataminer nalezla popis jména a nápovědy, je třeba tyto v souboru *Bundle.properties* uvést. Soubor *Bundle.properties* může tedy obsahovat mimo jiné například toto:

```
NAME_NazevModulu=Naive Bayes Classification
HINT_NazevModulu =Mining module for Naive Bayes Classification
```

Nyní se podrobněji podívejme na jednotlivé metody, které je nutné naimplementovat, aby se z komponenty stal plnohodnotný prvek dolovacího procesu. Všechny tyto metody jsou znázorněny i na obrázku 7.1.1.

run()

Tato metoda spustí akci, kterou má daná komponenta provádět. V našem případě, kdy je komponentou dolovací modul, bude tato metoda spouštět samotný dolovací algoritmus, tedy algoritmus pro naivní bayesovskou klasifikaci.

openCustomizingDialog()

Tato metoda zobrazí panel, který se má zobrazit po otevření dané komponenty. Pro náš modul bude tento panel obsahovat vstupní parametry a nastavení nutná pro daný algoritmus.

getOutputPanel()
Tato metoda má za úkol vytvořit panel, který se zobrazí, je-li na komponentu napojena komponenta Report. V našem případě tedy tato metoda bude vytvářet panel, který bude prezentovat výsledky získané samotným dolovacím modulem.

beforeAcceptEvent()
Tato metoda je volána před tím, než je komponenta vložena do grafu. Lze tedy nastavit například to, že komponenta může být v grafu dolovacího procesu pouze jedou.

afterAcceptEvent()
Tato metoda je volána po přidání komponenty do grafu. V našem případě tato metoda zapíše do DMSL souboru, že byla komponenta dolovacího modulu přidána do grafu.

afterRemoveEvent()
Metoda, která je volána po tom, co je komponenta odstraněna z grafu. V našem případě odstraní záznam v DMSL o komponentě dolovacího modulu.

7.2 Integrace vytvořeného modulu do aplikace

Nyní máme-li všechny potřebné metody naimplementovány a splnili jsme základní funkční požadavky, musíme ještě námi vytvořený modul do aplikace Dataminer zaintegrovat. To provedeme pomocí vytvoření záznamu v souboru *layer.xml*. Tento soubor představuje virtuální systém souborů, který se z různých modulů integruje do jednoho celku. To znamená, že záznamy, které tento modul vytvoří, budou viditelné i z ostatních modulů. Jednotlivé moduly o sobě vzájemně nic neví, nemají ani vzájemný přístup ke svým třídám a to i přesto, že běží v rámci jednoho virtuálního stroje Javy. O vytváření instancí se stará *System FileSystem*, který poskytuje instance ostatním modulům. [Krásný]. Pro integraci modulu do systému je nutné do souboru *layer.xml* přidat následující záznam:

```
<filesystem>
      <folder name="MiningPieceRegistry">
      <folder name="MiningModules">
            <file name="cz-vutbr-fit-dataminer-classifbcmodule-
            ClassifBCModule.instance">
            </file>
      </folder>
      </folder>
```

Tato část záznamu vytvoří instanci třídy modulu. A zaregistruje je v *MiningPieceRegistry* do kategorie *Mining Modules*, což povede k tomu, že komponenta bude zobrazována v paletě modulů zvané Mining Modules.

Zbylá část určuje množinu komponent, na které může být náš modul napojen a dále komponenty, které mohou být napojeny na tento modul. V našem případě *MiningPieceConfig* udává, že dolovací modul *ClassifBCModule* může být připojen pouze na komponentu *ReduceData*. Toto omezení plyne z toho, že komponenta redukce data dokáže data nejen redukovat, ale i rozdělit na data testovací a trénovací, což je pro funkčnost modulu pro naivní bayesovskou klasifikaci nezbytné. Bez tohoto rozdělení by modul nemohl fungovat. Dále tento záznam udává, ze napojena na tento modul může být jen komponenta sloužící ke zobrazení výsledků zvaná Report.

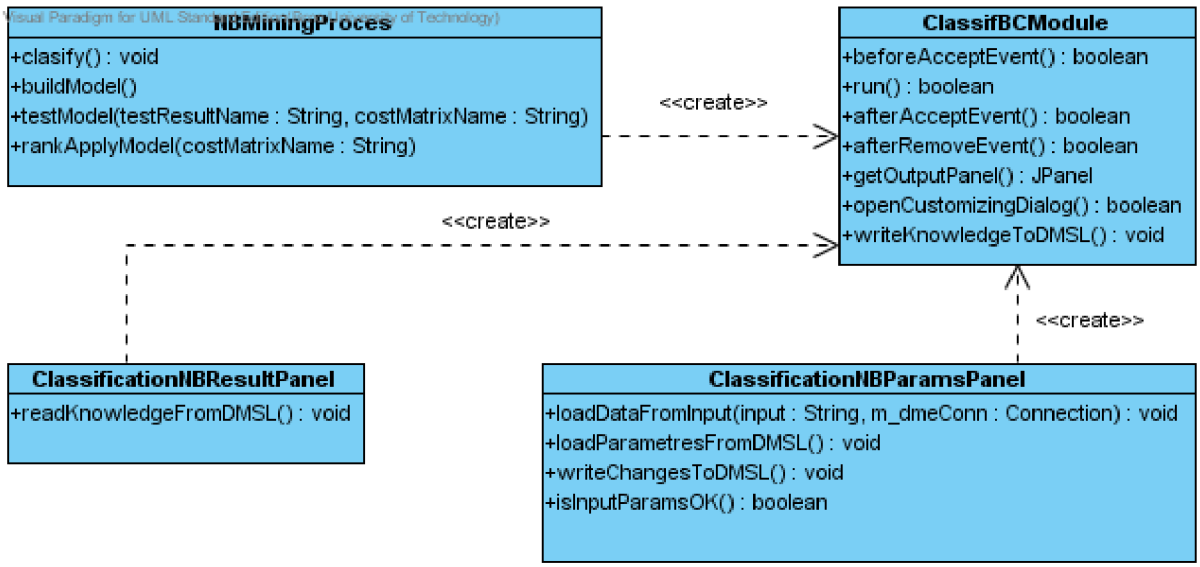
```
        <folder name="MiningPieceConfig">
        <folder name="cz-vutbr-fit-dataminer-classifbcmodule-
        ClassifBCModule">
        <folder name="accept">
            <file name="cz-vutbr-fit-dataminer-editor-palette-items-
            ReduceData"/>
        </folder>
        </folder>
        <folder name="cz-vutbr-fit-dataminer-editor-palette-items-
        Report">
        <folder name="accept">
             <file name="cz-vutbr-fit-dataminer-classifbcmodule-
             ClassifBCModule"/>
        </folder>
        </folder>
        </folder>
</filesystem>
```

8 Implementace modulu pro naivní bayesovskou klasifikaci

V této kapitole se budeme věnovat už samotné implementace daného klasifikačního modulu. Nejprve se budeme věnovat implementaci jednotlivých metod již zmiňované třídy *MiningPeace*. Dále se zmíníme o struktuře záznamu DMSL jak pro element *DataMiningTask*, tak i pro element *Knowledge*.

8.1 Implementace modulu

Hlavní část modulu pro naivní bayesovskou klasifikaci je implementována ve třídě *ClassifBCModule*, která rozšiřuje třídu *MiningPeace*. Třídou MiningPeace a implementací jejich metod se budeme zabývat v následující kapitole. Mimo tyto metody má modul za úkol načítání parametrů pro dolovací algoritmus z DMSL. Dále se stará o uložení získaného modelu a jeho testovacích metrik do DMSL. O to se stará metoda *writeKnowledgeToDMSL()*.



Obrázek 8.1.1 *Znázornění stěžejních tříd vytvořeného modulu*

8.2 Implementace metod děděných od třídy MiningPeace

V této části bude přiblížena implementace důležitých metod této třídy. Pro lepší orientaci budou metody řazeny v takovém pořadí, v jakém jsou volány v programu při vytváření grafu dolovacího procesu.

8.2.1 beforeAcceptEvent()

Jak již bylo řečeno, metoda je volána před tím, než je komponenta vložena do grafu. Je v ní pouze vygenerováno *id* modulu, aby jej šlo odlišit od ostatních komponent grafu.

8.2.2 afterAcceptEvent()

Přidáme-li do grafu dolovacího procesu další komponentu, kterou je právě modul pro naivní bayesovskou klasifikaci, je volána tato metoda. Po tomto vložení je nutné do DMSL záznam o tom, že do grafu byl vložen právě tento modul. V DMSL se tedy vytvoří následující záznam.

```
<DataMiningTask language="XML" name="DM582_ClassifBCModule1"
type="Clasification">
     <NBClassification algorithm="Naive Bayes">
          <InputInfo preferedTargetValue="null" primaryKey="null"
          target="null" />
          <Params>
               <SingletonTreshold>0.01</SingletonTreshold>
               <PairwiseTreshold>0.01</PairwiseTreshold>
               <NumberOfLiftQuantiles>10</NumberOfLiftQuantiles>
          </Params>
          <TargetProbabilities>
               <TargetValueProbability
               name="0">0.0</TargetValueProbability>
          </TargetProbabilities>
          <CostMatrix use="false" />
          <ROCResult use="true" />
          <LiftResult use="true" />
     </NBClassification>
</DataMiningTask>
```

Tímto tedy vytvoříme základ elementu *DataMiningTask* pro pozdější uložení všech potřebných parametrů. Hodnoty parametrů *SingletonTreshold*, *PairwiseTreshold* a *NumerOfLiftQuantiles* jsou automaticky nastaveny na výchozí hodnotu, tak jak je uvedeno v manuálu firmy *Oracle*. Uživatel je později může samozřejmě změnit.

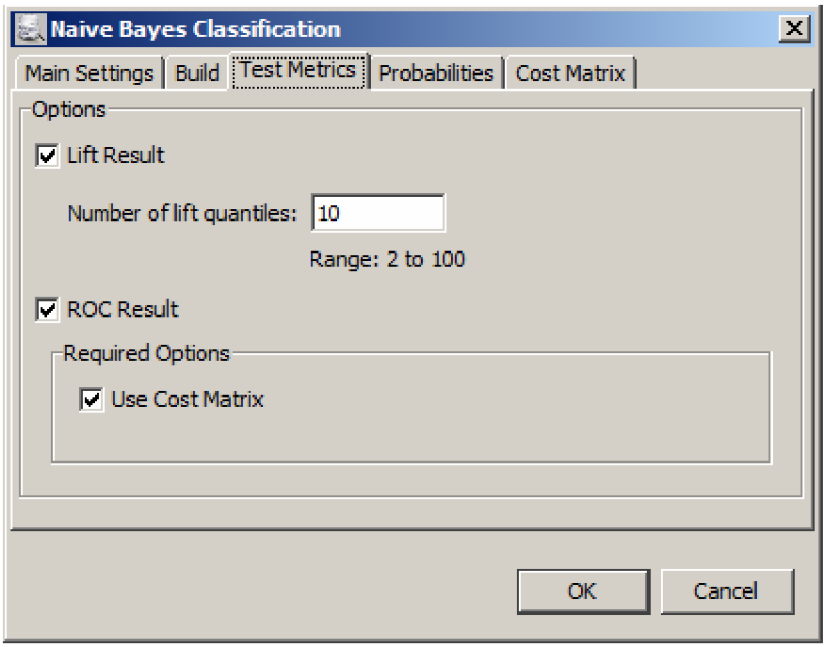
Dále se vytvoří základní záznam elementu *Knowledge*.

```
<Knowledge language="XML" name="DM582_ClassifBCModule1" />
```

Metoda *afterAcceptEvent*() tedy připraví DMSL záznam pro další úpravy, které proběhnou během samotného dolování.

8.2.3 openCustomizingDialog()

Tato metoda vytvoří a následně zobrazí panel, do kterého uživatel zadá potřebné parametry. Na panelu je pět záložek, ve kterých uživatel zadává jednotlivá nastavení pro samotný dolovací proces. Hlavní záložkou je *Main Settings*, která umožňuje zadání nezbytných parametrů, bez kterých nelze úspěšně dolovací proces spustit. V dalších záložkách jsou další nastavení, jako *pairwise treshold*, *singleton treshold*, *cost matrix*, probability table a další. Panel pro zadávání parametrů je zobrazen na obrázku 8.2.3.1.



Obrázek 8.2.3.1 *Ukázka panelu pro zadávání parametrů pro naivní bayesovskou klasifikaci*

Po zavolání metody *openCustomizingDialog()* jsou nejprve načteny potřebné informace, jako jsou názvy tabulek a jejich sloupců a následné naplnění comboboxů daného formuláře hodnotami z databáze. Pomocí nich poté uživatel nastavuje parametry pro dolovací proces. Dále má tato metoda k dispozici celý element *DataMiningTask* v DMSL souboru.

Po zadání všech parametrů a jejich potvrzení pomocí tlačítka „OK“ je obsah formuláře zkontrolován, zda zadané parametry odpovídají požadovanému formátu. Jsou-li chybně zadané, je uživatel upozorněn hláškou o nesprávném formátu vstupních parametrů. Je-li vše v pořádku, zadané parametry se uloží do DMSL. Implementace panelu pro vstupní parametry je ve třídě *ClassificationNBParamsPanel*.

8.2.4 run()

Tuto metodu Dataminer volá poté, co uživatel spustí komponentu modulu. Tato metoda spouští samotný dolovací proces. Nejprve ovšem načte všechny parametry z DMSL souboru. Tyto parametry se nepředávají přímo z formuláře pro zadávání parametrů, ale načítají se z DMSL souboru z jednoho hlavního důvodu. Tím je, že panel pro zadávání parametrů nemusí být vůbec před spuštěním dolování otevřen. Uživatel může pouze otevřít projekt uložený v DMSL a může rovnou spustit dolovací proces. V tomto případě, kdyby vstupní parametry byly předávány z již zmiňovaného formuláře, došlo by k chybě, protože by nebyly žádné parametry zadané. Po načtení všech parametrů jsou tyto předány dolovacímu algoritmu a ten je s nimi spuštěn.

Dalším vstupem mimo parametry jsou data. Tato data jsou uložena ve dvou databázových tabulkách. Tyto tabulky vzniknou jako výstup komponenty *Reduce/Partition*. Je proto nutné, aby tato komponenta předcházela komponentě dolovacího modulu pro naivní bayesovskou klasifikaci. Tato komponenta rozdělí data pro tvorbu modelu a na data testovací. Dolovací algoritmus je implementován ve třídě *NBMiningProces*.

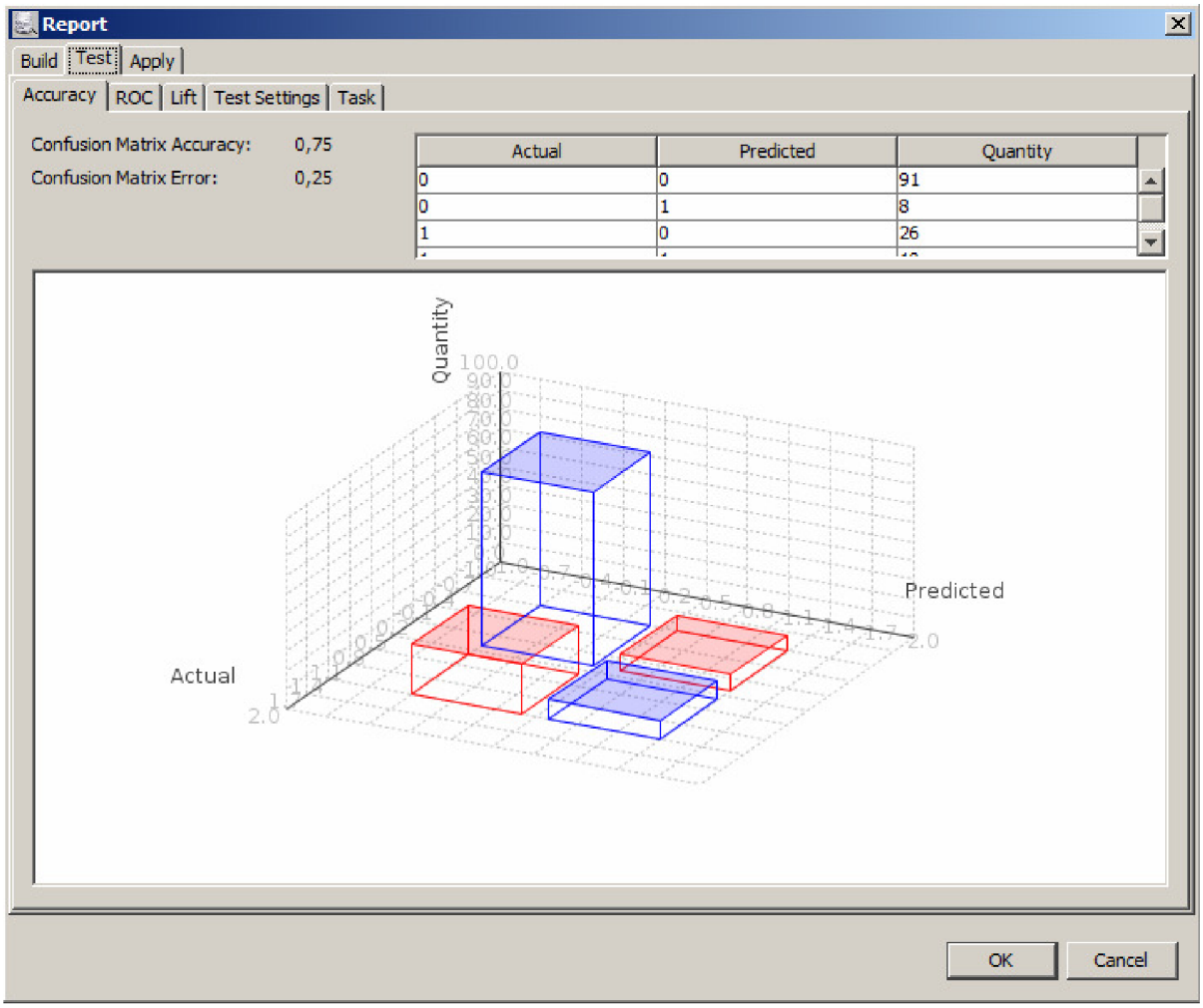
Algoritmus pracuje v následujících krocích. Pro připojení k databázi se inicializují všechny potřebné objekty a v databázi se vymažou případné tabulky, které jsou v ní uloženy z minulého spuštění algoritmu. Následně je vytvořen model podle nastavení, parametrů a dat, které uživatel zadal v panelu pro zadávání parametrů. Následně je tento model otestován na základě požadavků zadaných uživatelem. Jako volitelná část je aplikace daného modelu na konkrétní data a tudíž klasifikace

daných dat do jednotlivých tříd. Na závěr se veškeré informace o modelu a jeho metrikách uloží do DMSL. O formátu a způsobů uložení se zmíníme v následně.

Dolovací proces byl implantován za pomoci ukázkových příkladů firmy Oracle, kterým jsme se podrobněji věnovali v kapitole 6.

8.2.5 getOutputPanel()

Tato metoda vytvoří a zobrazí panel sloužící k prezentaci výsledků získaných během dolování. Obsahuje tři hlavní záložky a to *Build*, *Test* a *Apply*. V záložce *Build* může uživatel prohlížet samotný model pro klasifikaci. V záložce *Test* má možnost prohlížet jednotlivé metriky získané testováním daného modelu. Záložka *Apply* umožňuje prohlížení výsledků aplikace modelu na konkrétní data. Ukázka výstupního panelu je na obrázku 8.2.5.1. Podrobnější popis panelu s výsledky a panelu pro zadávání parametrů pro dolování bude uvedeno v kapitole s ukázkovým příkladem.



Obrázek 8.2.5.1 *Panel pro zobrazení výsledků – Confusion Matrix vizualizovaná pomocí JMathPlot*

8.2.6 afterRemoveEvent()

Metoda která je volána po tom, co je komponenta odstraněna z grafu. V našem případě odstraní z DMSL elementy *Knowledge* a *DataMiningTask*.

8.3 Použité externí knihovny

V této části práce bych rád zmínil použité knihovny, které byli použity při řešení tohoto projektu. Jedná se o knihovny, které vizualizují data z tabulek pomocí přehledných grafů. Nyní se postupně podíváme na obě z nich.

8.3.1 JFreeChart

JFreeChart je Java knihovna, která je zcela zdarma a je šířen pod *GNU Lesser General Public Licence (LGPL)* licencí. Tato knihovna umožňuje všem vývojářům vytvářet ve svých aplikacích profesionální grafy. Podporuje celou řadu vstupů, obsahuje širokou škálu grafů a má velice dobrou dokumentaci. Stejně tak má širokou škálu výstupů, od swingových komponent až po obrázkové soubory v různých formátech [JFreeChart]. Tato knihovna byla použita pro vykreslování 2D grafů. Konkrétně pak graf ROC křivky a grafy Liftů.

8.3.2 JMathPlot

JMathPlot je součástí projektu JMathTools. Což je kolekce nezávislých balíčků navržena pro běžné inženýrské a vědecké výpočty [JMathTools]. Pomocí JMathPlot je v tomto projektu vykreslován 3D graf Confusion Matrix.

8.4 Návrh elementů DMSL dokumentu

V této kapitole se seznámíme s návrhem a syntaxí elementů uchovávajících nastavení a získané znalosti pro klasifikační modul požívající naivní bayesovskou klasifikaci. Každý z vytvořených modulů musí mít vlastní návrh a definici DMSL elementů *DataMiningTask* a *Knowledge*. Je to z toho důvodu, že každý modul používá jiný algoritmus, pro který je třeba uchovávat různé parametry a každý modul vrací znalosti v jiné formě. Proto ani v rámci jádra systému nebyl návrh těchto dvou elementů proveden, protože jak již bylo řečeno, každý modul má jiné požadavky. Nyní si tedy popíšeme strukturu těchto elementů.

8.4.1 Element DataminingTask

Tento element slouží ke specifikaci dolovací úlohy. Syntaxe tohoto elementu není pevně daná, jediné dva povinné atributy jsou atributy *language* a *name*. Atribut *language* říká v jakém jazyce je element popsán a atribut *name* slouží jako vazba k danému dolovacímu modulu. Nyní si popíšeme syntaxi elementu pro uchování a nastavení parametrů pro naivní bayesovskou klasifikaci.

```
<!ELEMENT  DataMiningTask  ANY>
<!ATTLIST  DataMiningTask  name CDATA #REQUIRED
                           language CDATA #REQUIRED
                           type CDATA #IMPLIED
                           languageVersion CDATA #IMPLIED>
```

V elementu *DataMiningTask* je vložen element *NBClassification*, v rámci něj jsou uchovávány všechny potřebné parametry pro vytvořený modul.

```
<!ELEMENT NBClassification (InputInfo, Params, TargetProbabilities,
CostMatrix, ROCResult, LiftResult)>
```

```
<!ELEMENT InputInfo>
<!ATTLIST InputInfo  preferedTargetValue CDATA #REQUIRED
                     primaryKey CDATA #REQUIRED
                     target CDATA #REQUIRED>
```

V elementu *InputInfo* jsou uloženy informace o preferované cílové hodnotě třídy, primárním klíči tabulky, nad kterou se doluje a cílový atribut.

```
<!ELEMENT Params (SingletonTreshold, PairwiseTreshold,
NumberOfLiftQuantiles)>

<!ELEMENT SingletonTreshold (%REAL-NUMBER;)>
<!ELEMENT PairwiseTreshold (%REAL-NUMBER;)>
<!ELEMENT NumberOfLiftQuantiles (%INT-NUMBER;)>
```

K uložení pravděpodobností výskytu prvků v daných třídách slouží element *TargetProbabilities*, který obsahuje elementy *TargetValueProbability*.

```
<!ELEMENT TargetProbabilities (TargetValueProbability)>

<!ELEMENT TargetValueProbability (%REAL-NUMBER;)>
<!ATTLIST TargetValueProbability name CDATA #REQUIRED>
```

K uložení *cost matrix* slouží následující elementy.

Celá matice.
```
<!ELEMENT CostMatrix (CostMatrixRow)>
<!ATTLIST CostMatrix use (true|false) #REQUIRED>
```

Řádek matice.
```
<!ELEMENT CostMatrixRow (CostMatrixColumn)>
<!ATTLIST CostMatrixRow value CDATA #REQUIRED>
```

Buňka matice.
```
<!ELEMENT CostMatrixColumn (%REAL-NUMBER;)>
<!ATTLIST CostMatrixColumn position CDATA #REQUIRED>

<!ELEMENT ROCResult EMPTY>
<!ATTLIST ROCResult use (true|false) #REQUIRED>

<!ELEMENT LiftResult EMPTY>
<!ATTLIST LiftResult use (true|false) #REQUIRED>
```

8.4.2 Element Knowledge

Tento element slouží k uložení znalostí, které jsme získali během dolování spolu s informacemi, které jsou nutné pro správné zobrazení výsledků. Stejně jako u *DataMiningTask* syntaxe tohoto

elementu není pevně daná, jediné dva povinné atributy jsou atributy *language* a *name*. Atribut *language* říká v jakém jazyce je element popsán a atribut *name* slouží jako vazba k danému dolovacímu modulu.

```
<!ELEMENT  Knowledge  ANY>
<!ATTLIST  Knowledge  name CDATA #REQUIRED
                      language CDATA #REQUIRED
                      type CDATA #IMPLIED
                      languageVersion CDATA #IMPLIED>
```

V elementu *Knowledge* je vložen element *NaiveBayes*, v rámci něj jsou uchovávány všechny výsledky dolování a informace potřebné k jejich správné prezentaci.

```
<!ELEMENT NaiveBayes (ClassificationModel, TestMetrics, Apply)>

<!ELEMENT ClassificationModel (ModelSetting, TargetClass)>
```

Informace o vytvořeném modelu.

```
<!ELEMENT ModelSettings EMPTY>
<!ATTLIST ModelSettings    algorithmName CDATA #REQUIRED
                           endTime CDATA #REQUIRED
                           functionName CDATA #REQUIRED
                           pairwiseTreshold %REAL-NUMBER; #REQUIRED
                           singletonTreshold %REAL-NUMBER; #REQUIRED
                           startTime CDATA #REQUIRED
                           targetAtributName CDATA #REQUIRED
                           buildInput CDATA #REQUIRED
                           testInput CDATA #REQUIRED>
```

Uložení pravděpodobností modelu.

```
<!ELEMENT TargetClass (ProbabilityDetails)>
<!ATTLIST  TargetClass     name CDATA #REQUIRED
                           probability %REAL-NUMBER; #REQUIRED
                           value %INT-NUMBER; #REQUIRED
                           valueName CDATA #REQUIRED >
```

Jednotlivé pravděpodobnosti pro konkrétní hodnoty atributů.

```
<!ELEMENT ProbabilityDetails EMPTY>
<!ATTLIST ProbabilityDetails    probability %REAL-NUMBER; #REQUIRED
                                targetAtributName CDATA #REQUIRED
                                targetAtributValue CDATA #REQUIRED>
```

Elementy pro uložení testovacích metrik.

```
<!ELEMENT TestMetrics (TestMetricsSetting, CostMatrix,
ConfusionMatrix, Lift, ROC)>
```

```
<!ELEMENT TestMetricsSetting EMPTY>
<!ATTLIST TestMetricsSetting    endTime CDATA #REQUIRED
                                startTime CDATA #REQUIRED>
```

Elementy pro hodnoty Liftů.

```
<!ELEMENT Lift (LiftDetails, Quantile)>
<!ATTLIST Lift  use (true|false) #REQUIRED>

<!ELEMENT LiftDetails EMPTY>
<!ATTLIST LiftDetails      numberOfQuantiles %INT-NUMBER; #REQUIRED
                           positiveTargetValue CDATA #REQUIRED
                           targetAtributName CDATA #REQUIRED
                           totalCases %INT-NUMBER; #REQUIRED
                           totalPositiveCases %INT-NUMBER;
                           #REQUIRED>

<!ELEMENT Quantile EMPTY)>
<!ATTLIST Quantile    cumulativeLift %REAL-NUMBER; #REQUIRED
                cumulativeTargetDensity %REAL-NUMBER; #REQUIRED
                liftQuantile %REAL-NUMBER; #REQUIRED
                nonTargetsCumulative %INT-NUMBER; #REQUIRED
                percentageRecordsCumulative %REAL-NUMBER; #REQUIRED
                quantileNumber %INT-NUMBER; #REQUIRED
                quantileTargetCount %INT-NUMBER; #REQUIRED
                quantileTotalCount %INT-NUMBER; #REQUIRED
                targetDensity %REAL-NUMBER; #REQUIRED
                targetsCumulative %INT-NUMBER; #REQUIRED >
```

Elementy pro uložení hodnot pro ROC křivku.

```
<!ELEMENT ROC (ROCDetails, ROCRow)>
<!ATTLIST ROC   use (true|false) #REQUIRED>

<!ELEMENT ROCDetails EMPTY>
<!ATTLIST ROCDetails  numberOfTresholdCandidates %INT-NUMBER;
                      #REQUIRED
                      areaUnderCurve %REAL-NUMBER; #REQUIRED>

<!ELEMENT ROCRow EMPTY)>
<!ATTLIST ROCRow      falseNegatives %INT-NUMBER; #REQUIRED
                falsePositiveFraction %REAL-NUMBER; #REQUIRED
                falsePositives %INT-NUMBER; #REQUIRED
                index %INT-NUMBER; #REQUIRED
                probability %REAL-NUMBER; #REQUIRED
                trueNegatives%INT-NUMBER; #REQUIRED
                truePositiveFraction %REAL-NUMBER; #REQUIRED
                truePositives%INT-NUMBER; #REQUIRED>
```

Elementy pro *confusion matrix*.

```
<!ELEMENT ConfusionMatrix (ConfusionMatrixRow)>
<!ATTLIST ConfusionMatrix  accuracy %REAL-NUMBER; #REQUIRED
                           error %REAL-NUMBER; #REQUIRED
                           rowCount %INT-NUMBER; #REQUIRED>

<!ELEMENT ConfusionMatrixRow Empty>
<!ATTLIST ConfusionMatrix  actual %REAL-NUMBER; #REQUIRED
                           actualName CDATA #REQUIRED
                           prediction %REAL-NUMBER; #REQUIRED
                           predictionName CDATA #REQUIRED
                           value %INT-NUMBER; #REQUIRED>
```

Elementy pro aplikaci na konkretni data.

```
<!ELEMENT Apply (ApplySettings)>

<!ELEMENT ApplySettings EMPTY>
<!ATTLIST ApplySettings    startTime CDATA #REQUIRED
                           endtime CDATA #REQUIRED
                           tableName CDATA #REQUIRED
                           use (true|false) #REQUIRED
                           primaryKey CDATA #REQUIRED >
```

8.5 Problémy spojené s implementací

Při implementaci bylo nutné řešit celou řadu problémů. V této kapitole bych rád popsal ty, které byly nejzávažnější a jejich odstraní bylo časově nejnáročnější.

Zprovoznění sales history schématu

Tento problém nastal hned na počátku samotné implementace. Pro zpuštění ukázkových příkladů a následné testování implementovaného modulu bylo třeba mít přístup k datům, na kterých by bylo možné dolovat. Původně jsem sem se mylně domníval, že SH schéma na serveru musí být vytvořeno pouze administrátorem při jeho instalaci. K vyřešení problému s SH schématu mi napomohla konzultace s Ing. Rábem, který mi poskytnul i zdrojové skripty, které toto schéma vytvoří. Pak již bylo jen potřena nastavit správné jazykové prostředí v aplikaci a SH schéma bylo možné.

Zprovoznění ukázkového příkladu

Odstranění tohoto problému trvalo velice dlouho. Jednalo se o to, že když byl ukázkový problém spuštěn, došlo při běhu programu k řadě výjimek. V databázi se nevytvářely potřebné objekty a program tedy nebyl schopný korektně pracovat. Řešení jsem se pokoušel nalézt různými způsoby. Od znovuvytvoření SH schématu, přes úpravy zdrojových kódů ukázkového příkladu. Po dlouhé době hledání možných řešení jsem se rozhodl vyzkoušet spuštění dolovacího modulu na serveru pod jiným uživatelským účtem. Tímto se tento problém vyřešil. Problém byl v tom, že pro řešení diplomového projektu používám účet DMUSER1. Tento účet byl z počátku ovšem používán více studenty. Tomuto faktu přiřazuji i to, že to bylo důvodem, proč nastaly tyto problémy. Některý z dalších studentů mohl pozměnit databázi, čímž narušil její celkovou strukturu a program se poté s touto změnou nebyl schopný vypořádat. Požádal jsem tedy o resetování účtu a po znovuvytvoření SH schématu se problém již nevyskytoval a byl vyřešen.

Předání parametrů a vizualizace výsledků dolování

Při řešení projektu bylo nutné zajistit, aby dolovacímu algoritmu byly předány všechny potřebné parametry v požadovaném formátu. V ukázkovém příkladu nebyly používány všechny parametry, tudíž bylo třeba nastudovat, jaké jsou obecně všechny požadované parametry pro naivní bayesovskou klasifikaci.

Po implementaci dolovacího modulu bylo třeba získané znalosti vhodně vizualizovat. Největší problém byla vizualizace tzv. *confusion matrix*, ke jejíž vizualizaci bylo nejvhodnější použití 3D sloupcového grafu. Postupem času jsem dospěl k názoru, že nejlepší volbou bude použití externí knihovny *JMathPlot*, která nejlépe splňovala dané požadavky.

8.6 Možná rozšíření systému

Další dolovací moduly

Další možné rozšíření systému, které se okamžitě nabízí, je vytvoření dalších dolovacích modulů používajících jiné algoritmy. Souběžně s tímto modulem jsou vyvíjeny v rámci dalších diplomových prací ještě další moduly z oblasti klasifikace a predikce. Je ovšem možně vytvořit celou řadu dalších dolovacích modulů, které budou schopny spolupracovat s tímto systémem.

Porovnávání výsledků

Dále by bylo vhodné umožnit v rámci systému porovnávat jednotlivé dolovací algoritmy a jejich výsledky. To by bylo možné například pomocí nové komponenty, která by umožnila zobrazovat výsledky více dolovacích modulu v rámci jednoho panelu. V současném stavu systému je možné umístit do grafu dolovacího procesu pouze jednu komponentu Report a tu připojit pouze na jeden dolovací modul. Při zobrazení výsledků dvou modulů je tedy nutné tuto komponentu postupně připojovat k jednotlivým modulům, což je relativně nepohodlné a znesnadňuje vlastní porovnávání jednotlivých dolovacích metod.

Využití podpory jiných databázových prostředí

V současnosti systém využívá podpory Oracle. Bylo by možné systém upravit i pro použití nad jinými servery. Mohlo by se například jednat o server MySQL či jiný. To by s sebou samozřejmě přinášelo řadu úprav, které by bylo třeba provést. Byla by nutná úprava jak jádra systému, tak i jednotlivých dolovacích modulů, které jsou implementovány přímo pro Oracle a využívají podporu ODM.

9 Příklad možného použití modulu

V Této kapitole si ukážeme konkrétní příklad použití modulu pro naivní bayesovskou klasifikaci v aplikaci *Dataminer*. Popíšeme vše od počátku, tedy od vytvoření nového projektu, přes zadávání parametrů až po zobrazení výsledků.

V tomto příkladě použijeme data dodávaná firmou Oracle v již zmiňovaném *Sales History* schématu a budeme klasifikovat zákazníky do 2 skupin. A to zda jim bude, či nebude nabídnuta benefiční úvěrová karta (*affinity card*). V příloze na CD je přiložen SQL skript, který vytvoří v databázi tabulku *CUSTOMERS* obsahující všechna potřebná data, stejně tak je přiložen již vytvořený projekt, který obsahuje už veškeré nastavení a je jej možné otevřít a hned spustit dolovaní.

Nyní se tedy podívejme na jednotlivé kroky, které je nutné provést, abychom mohli začít dolovat z dat.

9.1 Vytvoření projektu v Datamineru

Po spuštění aplikace Dataminer je nutné vytvořit projekt, v rámci kterého budeme pracovat. To provodeme pomocí menu File -> New Project, případně pomocí klávesové zkratky Ctrl + Shift + N. Následně pomocí průvodce vybereme projekt *Data Mining Project* z kategorie *Knowledge Discovery* a klikneme na tlačítko Next. Dále potom nový projekt pojmenujeme a vybereme umístění jeho uložení a opět stiskneme tlačítko Next. Na závěr vybereme *New Oracle database connection* a nastavíme přihlašovací údaje k serveru. Funkční přihlašovací údaje jsou :

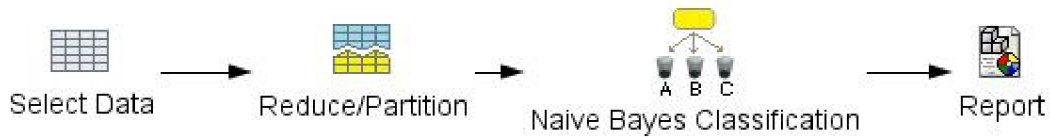
Login : dmuser1
Password : ondra1
Connection url : jdbc:oracle:thin:@berta.fit.vutbr.cz:1521:STUD

Na závěr již vše jen potvrdíme tlačítkem *Finish*. Tím jsme vytvořili prázdný projekt pro aplikaci Dataminer, který dále budeme upravovat.

9.2 Vytvoření grafu dolovacího procesu

Nyní je třeba vytvořit dolovací úlohu. To provedeme pomocí vytvoření grafu složeného z komponent z palety, která se nachází v pravé části okna Datamineru. Komponenty, které budeme nezbytně potřebovat, postupně umístíme na pracovní plochu, která se nachází uprostřed okna. Mezi nezbytné komponenty patří *Select Data*, *Reduce/Partition*, *Naive Bayes Classification* a *Report*. Ty je nutno propojit stejně jako je to uvedeno na obrázku 9.2.1.

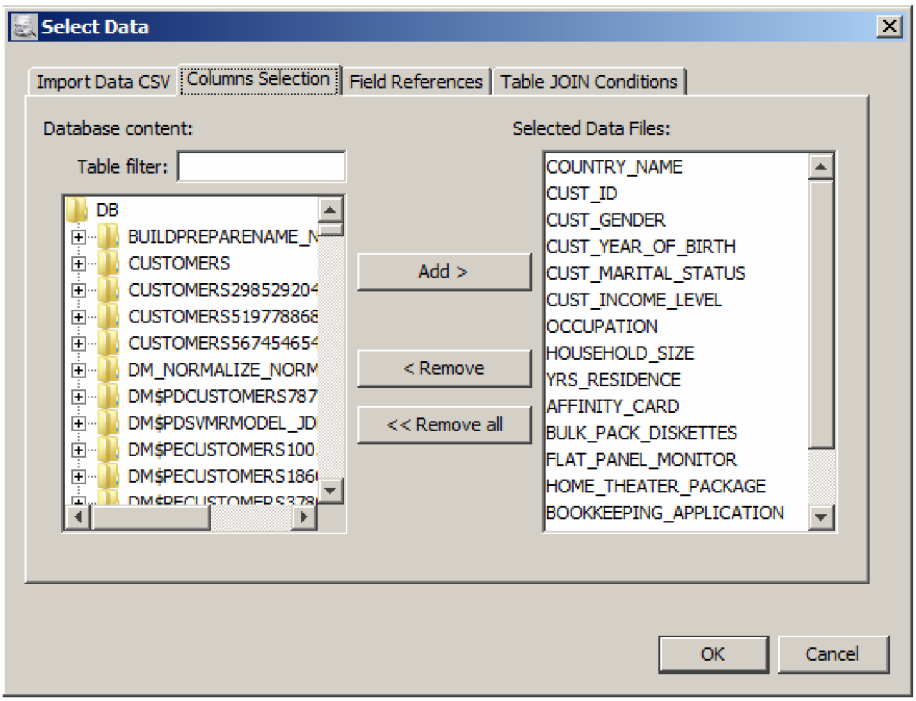
Kdybychom chtěli klasifikovat do tříd, které nejsou označeny čísly, jako je tomu v našem případě, museli bychom cílový atribut převést pomocí transformací na tzv. číselník. To lze provést pomocí komponenty Transformations tak, že cílový atribut pouze nezkopírujeme, ale místo možnosti *Copy* zvolíme *Create Refs*.



Obrázek 9.2. 1 *Možné zapojení komponent do grafu dolovacího procesu*

9.3 Výběr dat

K výběru dat slouží komponenta *Select Data*. Otevřeme-li tuto komponentu zobrazí se nám panel, pomocí, kterého můžeme zadat vstupní data pro dolování.

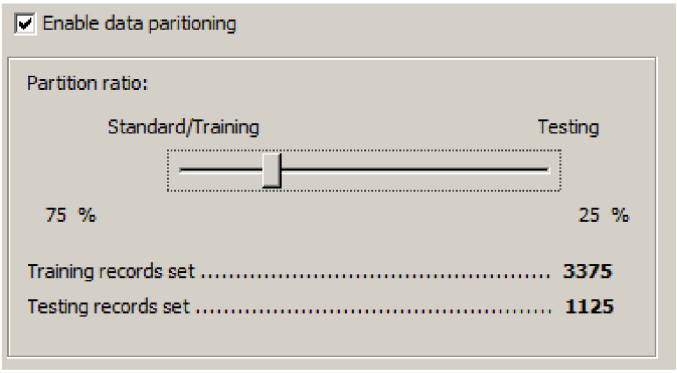


Obrázek 9.3. 1 *Panel komponenty select data aplikace Dataminer*

V tomto panelu vybereme záložku *Volume Selection* a poté v levé části okna vybereme tabulku CUSTOMERS a postupně pomocí tlačítka *Add* přidáme všechny atributy do *Selected Data Files*. Tím je pro tento příklad výběr dat ukončen. Tato komponenta umožňuje i řadu dalších funkcí, jako je spojování dat z více tabulek, importování dat ze souboru atd.

9.4 Rozdělení vstupních dat

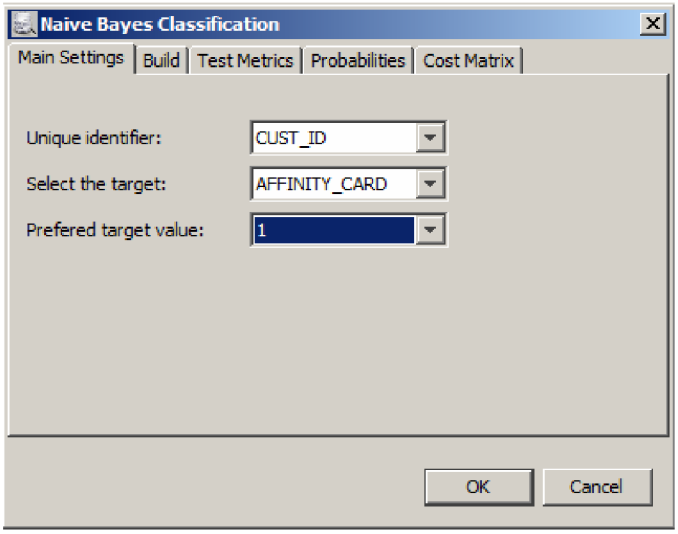
K rozdělení dat na data pro vytvoření modelu a data testovací slouží komponenta *Reduce/Partition*. Pomocí ní lze jednak data redukovat a v určitém poměru i rozdělit. Povolíme tedy rozdělení dat a to v poměru 75% Trénovací : 25% Testovací.



Obrázek 9.4. 1 *Rozdělení dat*

9.5 Nastavení parametrů pro dolování

Nyní se dostáváme k hlavní části, kdy musíme nastavit všechny potřebné parametry pro dolovací algoritmus, pro vytvoření modelu a jeho testování. Tyto parametry se zadávají po otevření komponenty dolovacího modulu *Naive Bayes Classification*. Po otevření této komponenty se nám zobrazí panel, který má 5 záložek. V záložce *Main Settings* vybereme primární klíč dané tabulky, tím je v našem případě atribut *CUST_ID*. Dále potom vybereme cílový atribut a tím je pro nás *AFFINITY_CARD* a dále pak preferovanou hodnotu tohoto atributu, tou je v našem případě hodnota 1. V ostatních záložkách je možno nastavit další parametry, jako jsou *singleton treshold*, *prior probalities* a další. Ty pro tento příklad můžeme nechat nastaveny defaultně. Jediné co ještě musíme udělat, je zaškrtnout checkbox *Use Cost Matrix* v záložce *Test Metrics*. Poté vše potvrdíme tlačítkem OK.



Obrázek 9.5. 1 *Nastavení parametrů*

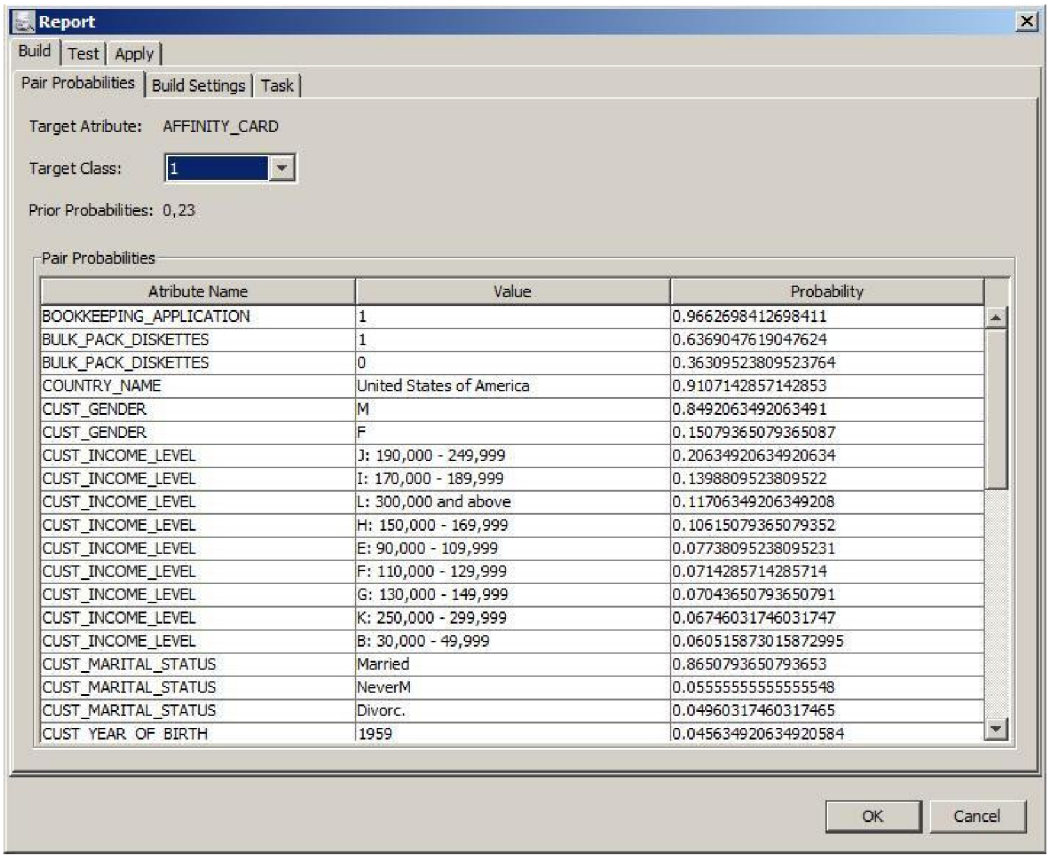
Nyní můžeme pustit dolovací proces. Ten spustíme tak, ze klikneme pravým tlačítkem myši na komponentu modulu a dáme run.

9.6 Zobrazení výsledků

K zobrazování výsledků slouží komponenta report. Po jejím otevření se nám v našem příkladě zobrazí panel, který má tři hlavní záložky. Jsou to *Build*, *Test* a *Apply*

9.6.1 Build

V této záložce se zobrazují informace o vytvořeném modelu. Záložka obsahuje další tři záložky. Konkrétně jsou to *Pair Probabilities*, *Build Setting* a *Task*. V záložce *Pair Probabilities* se nacházejí hodnoty atributů a jejich pravděpodobnosti na základě cílové třídy, do které se klasifikuje. Tak jak je to znázorněno na obrázku 9.6.1.1.



Obrázek 9.6. 1.1 *Záložka Pair Probabilities*

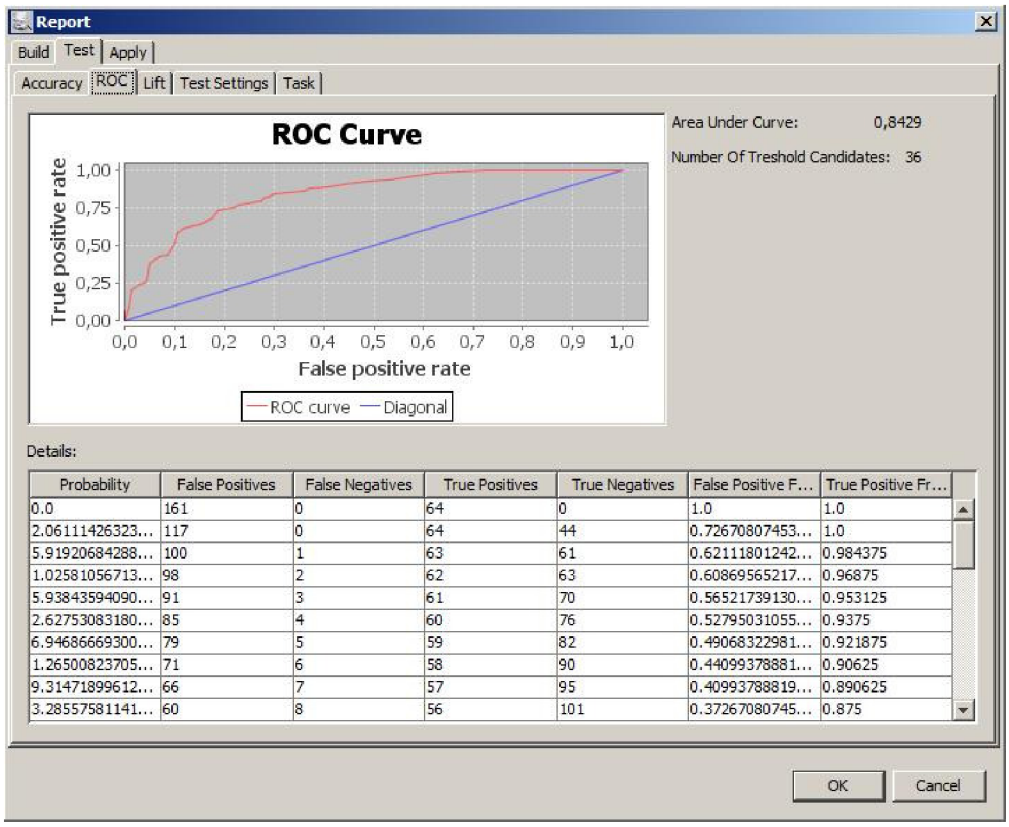
Z obrázku 9.6.1.1 lze tedy například zjistit, že má-li atribut zákazníka *BOOKKEEPING_APLICATION* hodnotu 1 je pravděpodobnost asi 0.966, že zákazník bude klasifikován do třídy *AFFININTY_CARD* = 1, tedy že mu bude nabídnuta benefiční karta. Analogicky pak tedy můžeme zjistit jednotlivé pravděpodobnosti pro ostatní hodnoty atributů.

V další záložce zvané *Build Setting* se nachází nastavení, se kterým byl daný model vytvořen. Uveden je název algoritmu, cílový atribut, pairwise treshold, singleton treshold a další. V poslední záložce jsou informace o samotné úloze vytvoření a to datum a čas začátku a konce vytváření modelu.

9.6.2 Test

Tato záložka obsahuje údaje, které jsme získali otestováním modelu, který jsme během dolování vytvořili a jeho zobrazení jsme se věnovali v předchozí kapitole. Záložka Test obsahuje další záložky. Konkrétně jsou to *Accuracy*, *ROC*, *Lift*, *Test Settings* a *Task*. Záložka *Accuracy* obsahuje informace o přesnosti daného modelu. Je v ní zobrazena tzv. *confusion matrix*, která udává počty klasifikovaných prvků do jednotlivých tříd a zároveň rozlišuje správně a špatně klasifikovaná testovací data. Jak vypadá záložka *Accuracy* je možné vidět na obrázku 8.2.5.1. 3D Graf v této záložce je vykreslován pomocí knihovny *JMathPlot*, která je zmíněna v kapitole 8.3.2.

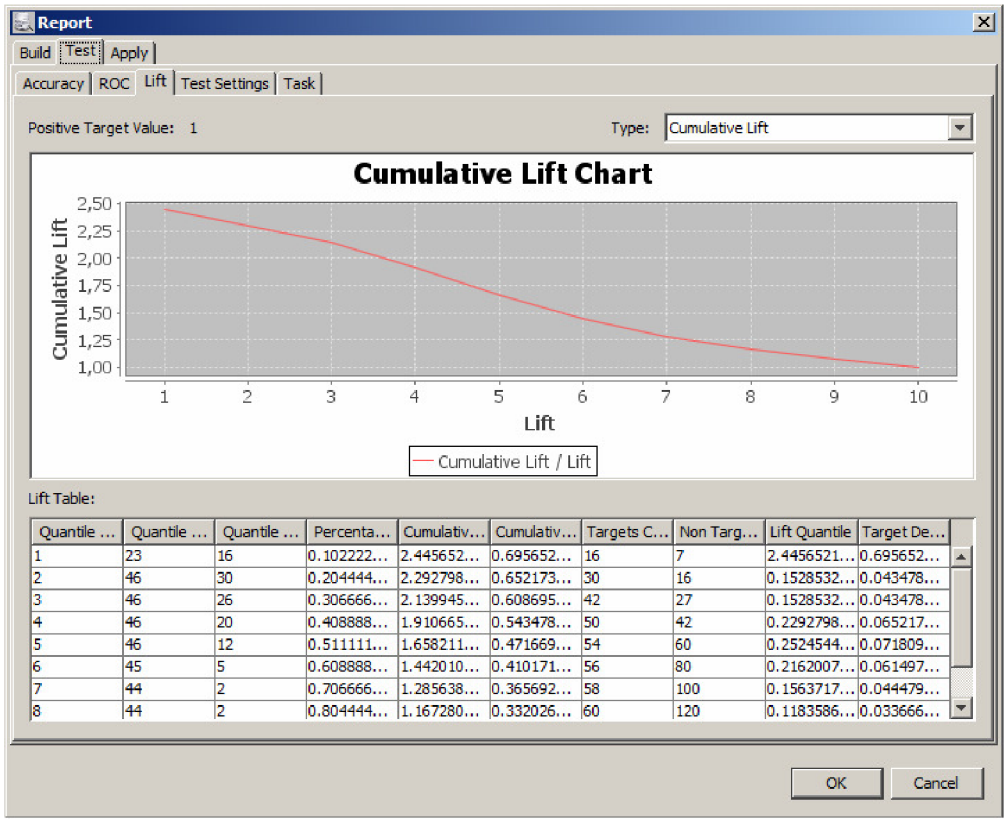
V záložce ROC je vizualizovaná ROC křivka. Spolu s grafem se v záložce nachází i tabulka s hodnotami, pomocí kterých je graf vykreslován a další informace jako například plocha pod křivkou atp. ROC křivka je vykreslována do grafu pomocí knihovny *JFreeChart*. Jak může vizualizace ROC křivky vypadat je znázorněno na obrázku 9.6.2.1.



Probability	False Positives	False Negatives	True Positives	True Negatives	False Positive F...	True Positive Fr...
0.0	161	0	64	0	1.0	1.0
2.06111426323...	117	0	64	44	0.72670807453...	1.0
5.91920684288...	100	1	63	61	0.62111801242...	0.984375
1.02581056713...	98	2	62	63	0.60869565217...	0.96875
5.93843594090...	91	3	61	70	0.56521739130...	0.953125
2.62753083180...	85	4	60	76	0.52795031055...	0.9375
6.94686669300...	79	5	59	82	0.49068322981...	0.921875
1.26500823705...	71	6	58	90	0.44099378881...	0.90625
9.31471899612...	66	7	57	95	0.40993788819...	0.890625
3.28557581141...	60	8	56	101	0.37267080745...	0.875

Obrázek 9.6. 2.1 *Záložka ROC*

Další záložkou je Lift. Ta obsahuje tabulku s daty a možnost zobrazovat různé druhy liftů. Jak taková záložka vypadá je zobrazeno na obrázku 9.6.2.2.



Quantile ...	Quantile ...	Quantile ...	Percenta...	Cumulativ...	Cumulativ...	Targets C...	Non Targ...	Lift Quantile	Target De...
1	23	16	0.102222...	2.445652...	0.695652...	16	7	2.4456521...	0.695652...
2	46	30	0.204444...	2.292798...	0.652173...	30	16	0.1528532...	0.043478...
3	46	26	0.306666...	2.139945...	0.608695...	42	27	0.1528532...	0.043478...
4	46	20	0.408888...	1.910665...	0.543478...	50	42	0.2292798...	0.065217...
5	46	12	0.511111...	1.658211...	0.471669...	54	60	0.2524544...	0.071809...
6	45	5	0.608888...	1.442010...	0.410171...	56	80	0.2162007...	0.061497...
7	44	2	0.706666...	1.285638...	0.365692...	58	100	0.1563717...	0.044479...
8	44	2	0.804444...	1.167280...	0.332026...	60	120	0.1183586...	0.033666...

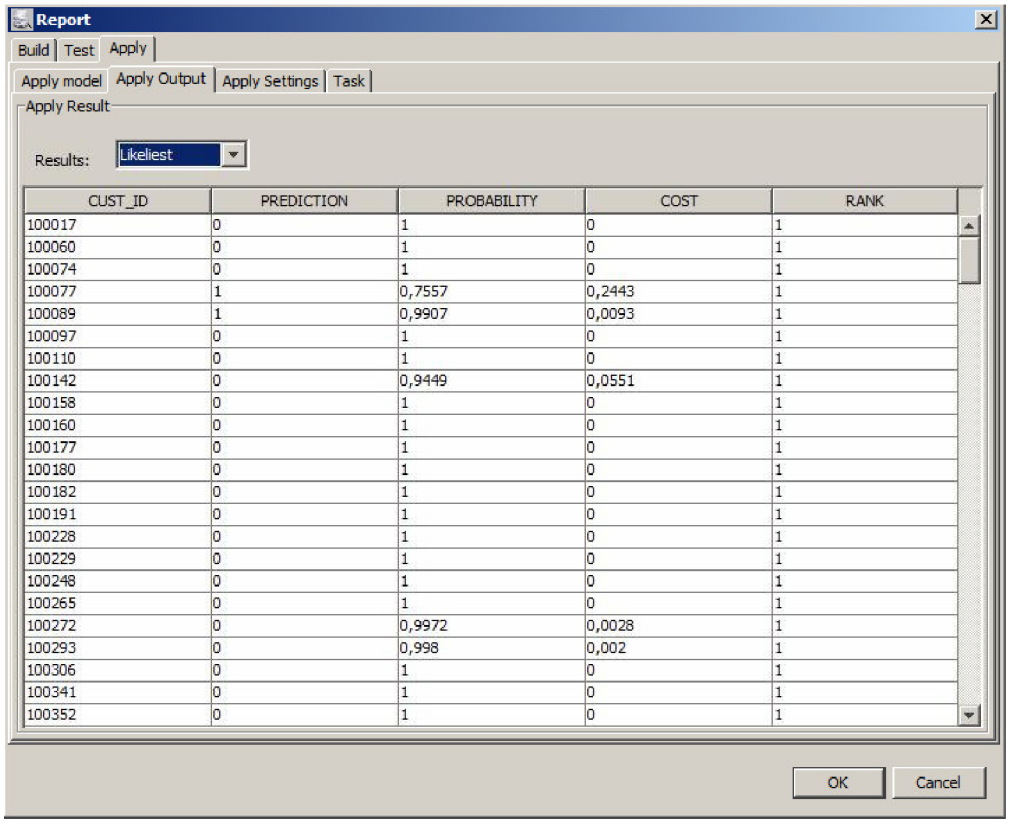
Obrázek 9.6. 2.2 *Záložka Lift*

Další záložkou je *Test Settings*. Ta obsahuje nastavení, se kterým byl daný model testován. Mezi tato nastavení patří informace o tom, zda se používá *ROC*, *Lift* a *Confusion Matrix* a dále potom Jaká byla použita ohodnocovací matice – tzv. *Cost Matrix*. Spolu s informací o použití liftů se zobrazuje i jejich počet.

Poslední záložkou je *Task*, ta obsahuje stejně jako u vytváření modelu informace o začátku a konci úlohy. Tedy datum a čas začátku a konce testování modelu.

9.6.3 Apply

V této záložce je možno aplikovat model na konkrétní data. Vstupní data lze zadat několika způsoby. Buďto vyplnit připravenou tabulku, nebo zadat název tabulky, která data obsahuje a je již uložena v databázi. Výstup potom může vypadat následovně.



CUST_ID	PREDICTION	PROBABILITY	COST	RANK
100017	0	1	0	1
100060	0	1	0	1
100074	0	1	0	1
100077	1	0,7557	0,2443	1
100089	1	0,9907	0,0093	1
100097	0	1	0	1
100110	0	1	0	1
100142	0	0,9449	0,0551	1
100158	0	1	0	1
100160	0	1	0	1
100177	0	1	0	1
100180	0	1	0	1
100182	0	1	0	1
100191	0	1	0	1
100228	0	1	0	1
100229	0	1	0	1
100248	0	1	0	1
100265	0	1	0	1
100272	0	0,9972	0,0028	1
100293	0	0,998	0,002	1
100306	0	1	0	1
100341	0	1	0	1
100352	0	1	0	1

Obrázek 9.6. 3.1 *Výsledek aplikace modelu na konkrétní data*

Jedná se tedy o tabulku, která obsahuje pravděpodobnosti, s jakými bude určitý prvek klasifikován do dané třídy. Lze zobrazovat všechny pravděpodobnosti pro všechny skupiny, nebo jen nejvyšší pravděpodobnosti pro skupinu, do které bude prvek klasifikován. Toto je možné provádět za pomocí roletkového menu *Result*. Volbou *All* zobrazíme všechny pravděpodobnosti, volbou *Likeliest* vybereme pouze nejvyšší pravděpodobnosti.

10 Závěr

Cílem tohoto diplomového projektu bylo po důkladné teoretické přípravě implementovat rozšiřující dolovací modul z oblasti klasifikace do systému pro dolování z dat, který je v současné době vyvíjen na FIT. Tato teoretická příprava se skládala z několika částí. V počátku bylo po nastudování teorie z oboru získávání znalostí z databází nutné seznámit se s již existující částí systému pro dolování z dat. Následně bylo třeba nastudovat a správně pochopit technologie, které se při vývoji tohoto systému používají, a dobře pochopit koncept dalšího rozšíření stávajícího stavu systému. V rámci své diplomové práce jsem se rozhodl implementovat klasifikační modul, který používá algoritmus pro naivní bayesovskou klasifikaci. Dále bylo třeba nastudovat práce kolegů, kteří tento systém postupně vyvíjeli a upravovali, aby na jejich práci bylo možné postupně navázat.

Stěžejní v této přípravné fázi bylo podrobné seznámení se s modulem Oracle Data Mining od firmy Oracle, který poskytuje podporu pro dolování z dat pro svou relační databázi. Konkrétně potom podpora pro naivní bayesovskou klasifikaci. V tomto ohledu byl velice vhodný ukázkový příklad, který dává firma Oracle k dispozici.

V rámci tohoto diplomového projektu byl tedy implementován přídavný dolovací modul pro naivní bayesovskou klasifikaci. Tento modul používá rozhraní, které poskytuje systém Dataminer. Modul plně využívá podpory ODM pro klasifikaci a umožňuje využít podporu pro algoritmus nivní bayesovské klasifikace. ODM umožňuje použití i jiných algoritmů, jako příklad lze uvést rozhodovací strom nebo SVM (Support Vector Machine).

Součástí tohoto diplomového projektu bylo nejen nové znalosti pomocí dolování získat, ale následně je i vhodně a přehledně prezentovat a vizualizovat. K tomu bylo použito přehledných tabulek, ale i 2D a 3D grafů, které velice usnadňují pochopení nové získaných znalostí. Spolu s implementací dolovacího modulu bylo nutné také navrhnout a vhodně rozšířit jazyk DMSL o popis elementů vhodných právě pro naivní bayesovskou klasifikaci tak, aby bylo možné uchovat informaci jak o vstupních parametrech, které jsou uchovávány v rámci elementu DataMiningTask, tak i uchování získaných znalostí v elementu Knowledge.

Jednoznačným přínosem této práce je rozšíření vyvíjeného systému o další funkčnost a tím i přiblížení jeho reálného nasazení do praxe a možnosti jeho začlenění do výuky namísto jiných komerčních produktů. Z toho vyplývá, že spolu s ostatními diplomovými projekty, které jsou řešeny paralelně s tímto by se ze systému měl stát plnohodnotný systém pro dolování z dat.

Tento projekt měl jednoznačný přínos i pro mou osobu. Prakticky jsem využil získané teoretické vědomosti, které jsem nabyl v rámci studia, stejně tak jsem získal řadu nových zkušeností a dovedností spojených se získáváním znalostí z databází. Bylo třeba vyřešit celou řadu problémů spojených s implementací, což také přispělo k získání podrobnějšího náhledu na problematiku dolování z dat.

Literatura

[Zendulka] Doc. Ing. Jaroslav Zendulka, CSc. a kol.: *Získávání znalostí z databází*. Brno, FIT VUT v Brně, 2006.

[Kmoščák] Ondřej Kmoščák: *Systém Vytvoření nových klasifikačních modulů v systému pro dolování z dat na platformě NetBeans,* semestrální projekt, Brno, FIT VUT v Brně, 2008

[Krásný] Michal Krásný: *Systém pro dolování z dat v prostředí Oracle,* diplomová práce, Brno, FIT VUT v Brně, 2008

[Mader] Pavel Mader: *Dolovací moduly systému pro dolování z dat v prostředí Oracle,* diplomová práce, Brno, FIT VUT v Brně, 2009

[Han] Jiawei Han, Micheline Kamber. *Data Mining: Concepts and Techniques.* Academic Press, 2001, San Diego, ISBN 1-55860-489-8

[Lukáš] Ing. Roman Lukáš, PhD. *Klasifikace a predikce-slajdy k předmětu ZZN*. Brno, FIT VUT v Brně.

[NetBeans] Vítejte u NetBeans a na stránkách www.netbeans.org. NetBeans, [online]. [cit. 3.9.2009]. Dostupné na URL: <http://www.netbeans.org/index_cs.html>

[Kotásek] Petr Kotásek: *DMSL: Data Mining Specification Language*, disertační práce, Brno, FIT VUT v Brně, 2003.

[Doležal] Jindřich Doležal : *Jádro systému pro dolování z dat v prostředí Oracle*, diplomová práce, Brno, FIT VUT v Brně, 2006.

[Gálet] Michal Gálet: *Grafická nádstavba pro systém získávání znalostí*, diplomová práce, Brno, FIT VUT v Brně, 2006.

[Oracle a] Oracle Data Mining. Oracle, [online]. [cit. 22.4.2009]. Dostupné na URL: <http://www.oracle.com/global/cz/database/data_mining.html >

[Oracle b] Oracle SQL Developer. Oracle, [online]. [cit. 23.4.2009]. Dostupné na URL: <http://www.oracle.com/technology/products/database/sql_developer/index.html>

[Oracle c] Naive Bayes. Oracle, [online]. [cit. 23.4.2009]. Dostupné na URL: <http://download.oracle.com/docs/cd/B28359_01/datamine.111/b28129/algo_nb.htm#DMCON018>

[JFreeChart] Welcome To JFreeChart!. JFreeChart, [online]. [cit. 8.5.2009]. Dostupné na URL: < http://www.jfree.org/jfreechart/ >

[JMathTools] What JMathTools IS. JMathToolst, [online]. [cit. 8.5.2009]. Dostupné na URL: < http://jmathtools.berlios.de/doku.php>

Seznam příloh