VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

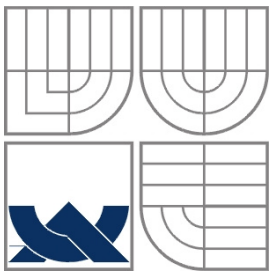
DOLOVACÍ MODUL SYSTÉMU PRO DOLOVÁNÍ Z DAT NA PLATFORMĚ NETBEANS

DIPLOMOVÁ PRÁCE
MASTER‘S THESIS

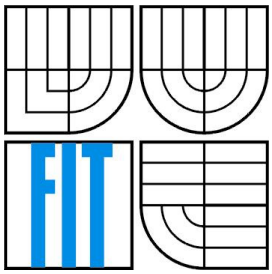
AUTOR PRÁCE
AUTHOR

Bc. JAROMÍR VÝTVAR

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

DOLOVACÍ MODUL SYSTÉMU PRO DOLOVÁNÍ Z DAT NA PLATFORMĚ NETBEANS

DATA MINING MODULE OF A DATA MINING SYSTEM ON NETBEANS PLATFORM

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. JAROMÍR VÝTVAR
AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. JAROSLAV ZENDULKA, CSc.
SUPERVISOR

BRNO 2010

Zadání diplomové práce/10334/2009/xvytva00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2009/2010

Zadání diplomové práce

Řešitel: **Výtvar Jaromír, Bc.**
Obor: Informační systémy
Téma: **Dolovací modul systému pro dolování z dat na platformě NetBeans**
Data Mining Module of a Data Mining System on NetBeans Platform
Kategorie: Databáze

Pokyny:

1. Seznamte se s problematikou získávání znalostí z databází. Seznamte se s aktuální implementací jádra systému a již vytvořenými dolovacími moduly.
2. Po dohodě s vedoucím diplomové práce zvolte typy dolovaných znalostí a algoritmy, pro něž vytvoříte pro systém dolovací modul. Zvažte možnost využití dostupných implementací dolovacích algoritmů.
3. Zvolený modul implementujte, integrujte se zbytkem systému a ověřte funkčnost.
4. Zhodnoťte dosažené výsledky.

Literatura:

- Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Second Edition. Elsevier Inc., 2006, 770 p.
- Šebek, M.: Rozšíření funkcionality systému pro dolování z dat na platformě NetBeans. Diplomová práce. FIT VUT v Brně. 2009.
- Mader, P.: Dolovací moduly systému pro dolování z dat v prostředí Oracle. Diplomová práce. FIT VUT v Brně. 2009.
- Henkl, T.: Dolovací moduly systému pro dolování z dat na platformě NetBeans. Diplomová práce. FIT VUT v Brně. 2009.
- Oracle Data Mining. Dokumentace v online knihovně Oracle. Dostupné na http://www.oracle.com/pls/db102/homepage.

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: **Zendulka Jaroslav, doc. Ing., CSc.**, UIFS FIT VUT
Datum zadání: 21. září 2009
Datum odevzdání: 26. května 2010

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

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Cílem této diplomové práce je získat přehled o procesu získávání znalostí z databází a analýza dolovacího systému vyvíjeného na FIT VUT v Brně na platformě NetBeans za účelem vytvoření nového dolovacího modulu. Ze získaných znalostí bylo rozhodnuto o vytvoření modulu pro dolování odlehlých hodnot a doplnění existujícího modulu regrese o nový algoritmus vícenásobné lineární regrese založený na zobecněných lineárních modelech. Nové dolovací metody využívají existující řešení na straně Oracle data mining.

Abstract

The aim of this work is to get basic overview about the process of obtaining knowledge from databases – datamining and to analyze the datamining system developed at FIT BUT on the NetBeans platform in order to create a new mining module. We decided to implement a module for mining outliers and to extend existing regression module with multiple linear regression using generalized linear models. New methods using existing methods of Oracle Data Mining.

Klíčová slova

Získávání znalostí z databází, dolování dat, vícenásobná regrese, predikce, odlehlé hodnoty, Oracle data mining, Java, NetBeans, Support vector machine, zobecněné lineární modely.

Keywords

Knowledge discovery in databases, data mining, multiple regresion, predicting, outliers, Oracle data mining, Java, NetBeans, Support vector machine, generalized linear models.

Citace

Výtvar Jaromír: Dolovací modul systému pro dolování z dat na platformě NetBeans,
diplomová práce, Brno, FIT VUT v Brně, 2010.

Dolovací modul systému pro dolování z dat na platformě NetBeans

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana doc. Ing. Jaroslava Zendulky, CSc.
Další informace mi poskytl pan Ing. Michal Šebek.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Jaromír Výtvar
25. května 2010

Poděkování

Rád bych poděkoval panu doc. Ing. Jaroslavu Zendulkovi, CSc. za vstřícné kroky a rady při výběru zaměření a řešení této diplomové práce. Dále bych chtěl poděkovat panu Ing. Michalu Šebkovi za poskytnutí užitečných informací k vyvíjenému systému.

Obsah

Obsah 1
1 Úvod 4
2 Získávání znalostí z databází 6
2.1 Proces získávání znalostí 6
2.2 Druhy dat pro dolování 7
2.3 Příprava a předzpracování dat 8
2.3.1 Čištění dat 8
2.3.2 Integrace dat 10
2.3.3 Výběr dat 10
2.3.4 Transformace dat 10
2.4 Typy dolovacích úloh a algoritmy jejich řešení 11
2.4.1 Popis konceptu/třídy 11
2.4.2 Frekventované vzory a asociační analýza 11
2.4.3 Klasifikace a predikce 12
2.4.4 Shluková analýza 13
2.4.5 Dolování odlehlých hodnot 14
2.4.6 Evoluční analýza 14
2.5 Vyhodnocení získaných výsledků 14
3 Predikce 16
3.1 Lineární jednoduchá regrese 17
3.2 Lineární vícenásobná regrese 18
3.3 Nelineární regrese 20
3.4 Zobecněné lineární modely 20
3.4.1 Podmínky klasického lineárního modelu 20
3.4.2 Zobecněný lineární model 22
4 Dolování odlehlých hodnot 23
4.1 Příprava a předzpracování dat 23
4.2 Metody dolování odlehlých hodnot 24
4.2.1 Z pohledu statistických měr 24
4.2.2 Z pohledu dolovacích úloh 26
4.3 Support vector machines – SVM 30
5 Dataminer 32
5.1 Platforma NetBeans 32
5.2 Oracle Data Mining 32

5.3 DMSL 33
5.4 Základní struktura systému 35
5.4.1 Jádro systému 35
5.4.2 Grafické uživatelské rozhraní 35
5.4.3 Dolovací moduly 36
5.5 Historie a současný stav 37
5.6 Navrhovaná rozšíření 37
5.6.1 Rozšíření o dolovací metody z ODM 38
5.6.2 Rozšíření o další dolovací metody 38
5.7 Plánovaná rozšíření v této práci 38
6 Nový modul detekce odlehlých hodnot 39
6.1 Vytvoření a integrace modulu do systému 39
6.2 Algoritmus one-class SVM v ODM 40
6.2.1 Nastavitelné parametry 40
6.2.2 Předzpracování dat 41
6.2.3 Rozhraní ODM pro práci s one-class SVM 41
6.3 Úpravy v DMSL 43
6.3.1 Element DataMiningTask 43
6.3.2 Element Knowledge 44
6.4 Implementace modulu 45
6.4.1 Implementace třídy MiningPiece 46
6.4.2 Struktura modulu 47
6.4.3 Vložení komponenty modulu 48
6.4.4 Panel parametrů 48
6.4.5 Spuštění dolování 49
6.4.6 Prezentace výsledků 49
7 Rozšíření modulu predikce 51
7.1 Algoritmus GLM v ODM 51
7.1.1 Nastavitelné parametry 51
7.1.2 Předzpracování dat 52
7.1.3 Rozhraní ODM pro práci s GLM 52
7.2 Rozšíření DMSL 54
7.2.1 Element DataMiningTask 54
7.2.2 Element Knowledge 55
7.3 Implementace rozšíření modulu 56
7.3.1 Struktura rozšířeného modulu 56
7.3.2 Vložení komponenty modulu 57

7.3.3 Panel parametrů 57
7.3.4 Spuštění dolování 58
7.3.5 Prezentace výsledků 58
8 Příklady použití modulů 59
8.1 Modul predikce - GLM 59
8.1.1 Nadefinování komponent v grafu dolovacího procesu 59
8.1.2 Vstupní data, parametry a vytvoření modelu 60
8.1.3 Prezentace výsledků 60
8.1.4 Aplikační fáze 62
8.2 Modul detekce odlehlých hodnot 62
8.2.1 Nadefinování komponent dolovacího procesu 63
8.2.2 Vstupní data, parametry a vytvoření modelu 63
8.2.3 Prezentace výsledků 63
8.2.4 Aplikační fáze 64
9 Závěr 66

1 Úvod

V moderním současném světě produkuje každý člověk, společnost i příroda každým dnem, hodinou či minutou nová data. Tato veškerá data, která nyní celosvětově vznikají, jsou ukládána především do databází jednotlivých společností, státních úřadů, jednotlivých vlád, zdravotnických zařízení i jednotlivců. Objemy těchto dat jsou značně velké a zahrnují v sobě netriviální popisy a vazby, tedy informace o chování různých procesů, zařízení a lidí na celém světě. Znalosti těchto informací slouží především pro podporu strategického rozhodování. V současnosti jsou vzhledem k probíhající krizi stále více požadované a ceněné, protože bez dostatečných informací nelze vytvářet dobrá a perspektivní rozhodnutí. Současně také tyto informace skrývají dosud neobjevené souvislosti posouvající poznání celého lidstva o další krok dále.

Získáním těchto informací se zabývá odvětví informačních technologií nazvané získávání znalostí z databází, anglicky knowledge discovery in databases – zkráceně KDD nebo také dolování dat, anglicky data mining. Samotné získávání znalostí z databází je složitý proces, skládající se z mnoha kroků a obsahující náročné algoritmy nad velkými objemy dat. Cílem je tento proces plně automatizovat tak, aby byl schopen dolovat znalosti nad různými druhy vstupních dat s využitím nejvhodnějších algoritmů a ziskem co nejlepších výsledků.

V současnosti tak vzniká mnoho nástrojů pro dolování dat řešících různé dolovací úlohy jak v akademickém prostředí vysokých škol, tak i v čistě komerční sféře. A to jak na úrovni samostatných programů, tak i podpory dolování v již existujících databázových aplikacích. Vzhledem k neustálému vývoji v tomto odvětví se tyto nástroje stále vylepšují a doplňují. Na Fakultě informačních technologií Vysokého učení technického v Brně je také již od roku 2002 vyvíjen dolovací nástroj Dataminer. Tento nástroj je koncipován jako experimentální všestranný dolovací nástroj, který se snaží pokrýt všechny dolovací úlohy nad různými daty pro dolování.

Tato práce si klade za úkol zdokumentovat současný stav vývoje Datamineru, zkompletovat informace o již existujících dolovacích modulech a použitých algoritmech a navrhnout další možnosti jeho rozšíření o nové moduly či algoritmy. V konečné fázi je cílem práce výběr a implementace jednoho nového modulu pro detekci odlehlých hodnot a rozšíření již existujícího modulu regrese o vícenásobnou lineární regresi využívající algoritmus zobecněných lineárních modelů a následně jejich začlenění do Datamineru.

Text práce je strukturován do devíti kapitol. Úvodní kapitola seznamuje s obecným pojmem dolování dat v širším kontextu a strukturou práce. V druhé kapitole je podrobnější seznámení s oborem získávání znalostí z databází od obecného procesu dolování dat až po konkrétní typy dolovacích úloh a příklady algoritmů pro jejich řešení. Zvláštní důraz je kladen na dolovací metody predikce a dolování odlehlých hodnot, uvedené ve třetí a čtvrté kapitole. Pátá kapitola je zaměřena na popis dolovacího nástroje Datamineru, na současný stav jeho vývoje a na možnosti jeho rozšíření

zejména v oblasti dolování dat s podporou Oracle data mining – ODM. Šestá kapitola je věnována návrhu a implementaci nového dolovacího modulu detekce odlehlých hodnot pro nástroj Dataminer a sedmá kapitola popisuje návrh a implementaci rozšíření již existujícího dolovacího modulu predikce, který je již obsažen v nástroji Dataminer, o nový dolovací algoritmus založený na zobecněných lineárních modelech. Osmá kapitola pak ukazuje způsob práce s implementovanými moduly a prezentaci vydolovaných znalostí. Poslední závěrečná kapitola shrnuje celou tuto práci a dosažené výsledky.

2 Získávání znalostí z databází

Nejprve si definujeme pojem získávání znalostí z databází. Následující definice je nejčastěji používána a nejlépe vystihuje celou problematiku. *Získávání znalostí z databází* je extrakce (neboli „dolování“) zajímavých (netriviálních, skrytých, dříve neznámých a potenciálně užitečných) modelů dat a vzorů z velkých objemů dat. Tyto modely a vzory reprezentují znalosti získané z dat [1].

Pro správné pochopení je důležité vysvětlit si, že *netriviální* a *skrytý* znamená jednoduše nezískatelný a na první pohled neviditelný model nebo vzor. Jako příklad lze uvést informaci nezískatelnou pouhým klasickým SQL dotazem nad databází. Je tedy nutné využít složitějších postupů a algoritmů. Získané znalosti také musí být *potenciálně užitečné*, tedy takové, aby posloužily zadavateli k podpoře rozhodování nebo potvrzení nějaké domněnky. Za získávání znalostí z databází nejsou považovány ani expertní nebo deduktivní databázové systémy.

Jako klasický příklad lze použít prodejnu potravin. Každý nákup je ukládán do databáze pro účely účetní a statistické. Lze tedy jednoduše získat informace o počtu prodaných rohlíků a celkové částce utržených peněz za ně. Netriviální informací pro majitele obchodu může být například, pokud si zákazník koupil rohlík, tak si k němu s vysokou pravděpodobností koupí i máslo, a pokud si koupí rohlík i máslo, celkem s jistotou si koupí i noviny. Je pro něj tedy přínosné, má-li takových zákazníků většinu, dát do blízkosti rohlíků stojan s máslem a nerušit prodej novin.

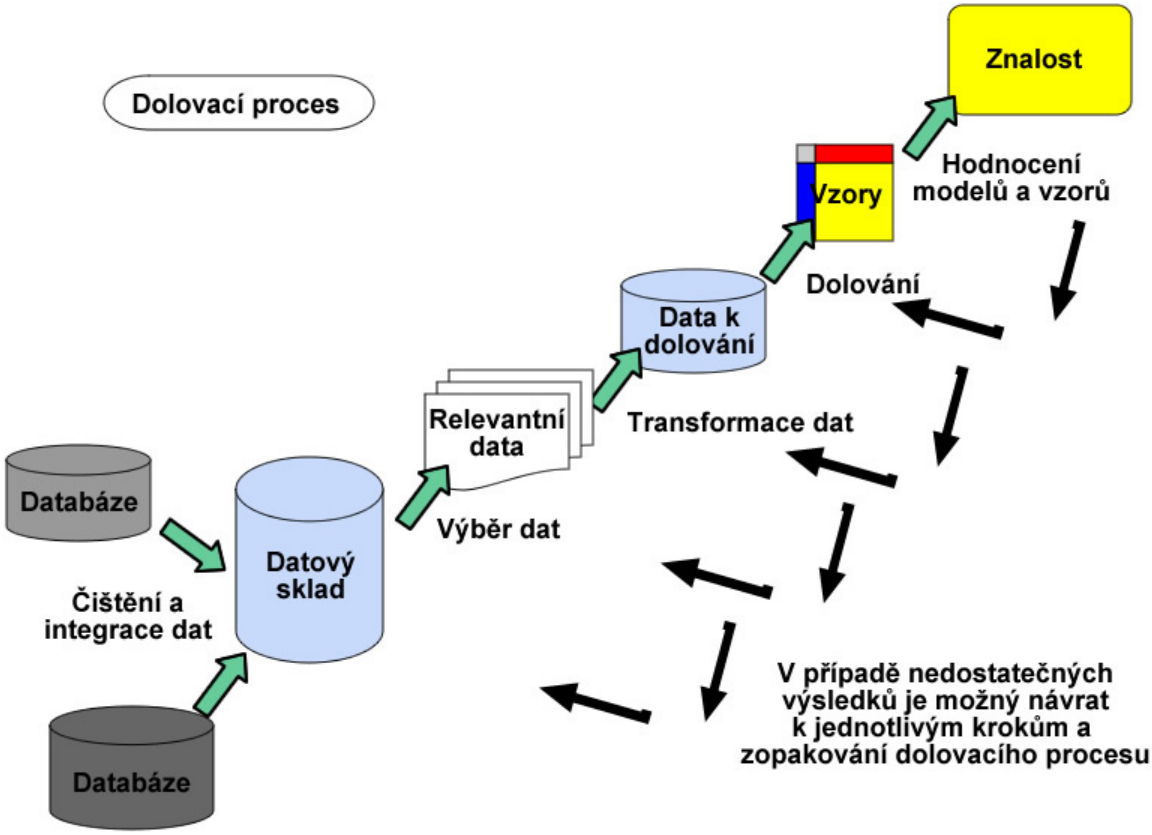
2.1 Proces získávání znalostí

Získávání znalostí z databází je ucelený proces obsahující na sebe navazující kroky, které jsou popsány na obrázku 2.1 a následně podrobně rozepsány.

Podrobný popis jednotlivých kroků:

1. **Čištění dat** – odstraní nekonzistence v tabulkách, vhodně upraví nevyplněné hodnoty a opraví zašuměná data.
2. **Integrace dat** – sloučí všechny potřebné tabulky pocházející z různých zdrojů a zajistí konzistence mezi tabulkami.
3. **Výběr dat** – vybere všechna relevantní data potřebná pro dolovací úlohu. Dojde tak k odfiltrování nepodstatných dat a zmenšení objemu zpracovávaných dat.
4. **Transformace dat** – upraví data do vhodného formátu pro dolování. Například může proběhnout sumarizace, agregace nebo rozdělení některých sloupců tabulek.
5. **Dolování dat** – je samotný proces získání vzorů a modelů pomocí daného dolovacího algoritmu.
6. **Hodnocení modelů a vzorů** – identifikuje pouze přínosné nebo zajímavé vzory.

7. **Prezentace znalostí** – znamená konečnou úpravu vydolovaných znalostí do vizuální formy a interpretaci získaných výsledků tak, aby byly pochopitelné pro koncového uživatele.



Obrázek 2.1 Proces získávání znalostí z databází

Celý proces se také dá rozdělit na přípravu a předzpracování dat, dolovací úlohu a vyhodnocení získaných znalostí. Část přípravy a předzpracování dat obsahuje první čtyři kroky. Následuje dolovací úloha, tedy pátý krok a vyhodnocení získaných výsledků obsahující poslední dva kroky.

2.2 Druhy dat pro dolování

Obecně lze dolovat v jakémkoliv druhu dat. Lze provést základní dělení na data perzistentní a transientní. Perzistentní data jsou uložena na nějakém úložišti. Transientní data nejsou v době zpracování nikde uložena, ale průběžně přicházejí. Lze je také pojmenovat jako proudy dat.

Jako příklady perzistentních dat můžeme uvést různé typy databází:

- **Relační databáze** – nejčastěji používaný zdroj dat, kde dotazy jsou kladeny klasickým jazykem SQL.
- **Datové sklady** – vznikají za účelem analytického dotazování nad velkým množstvím dat z menších relačních databází, z těchto důvodů jsou data organizována kolem

hlavních dotazovaných subjektů, nejčastěji bývají modelována jako multidimenzionální datové struktury.

- **Multimediální databáze** – slouží k ukládání multimediálních dat (obrázek, audio, video).
- **Textové databáze** – obsahují strukturované, částečně strukturované i nestrukturované texty, například velmi rozsáhlé databáze knihoven.
- **Heterogenní databáze** – spojují různé druhy dat v různých typech databází.
- **Web** – je velká heterogenní databáze obsahující text a multimediální obsah, velký prostor pro dolování dat.
- **Transakční databáze** – obsahují záznamy jednotlivých transakcí.
- **Objektově-relační databáze** – obsahují uložené objekty a umožňují složitější struktury než relační databáze.
- **Temporální databáze** – jsou určené pro ukládání a práci s daty v čase a udržují historii změn.
- Další používané databáze: databáze sekvencí, databáze časových řad, prostorové databáze, prostorově-temporální databáze.

Příklady transientních dat:

- **Proudy dat** – souvislé datové toky například z multimediálních zařízení, kamerových systémů, odposlouchávacích zařízení, síťové komunikace. Zatím málo prozkoumaná oblast dolování dat.

2.3 Příprava a předzpracování dat

Příprava a předzpracování dat je důležitou částí procesu získávání znalostí z databází. Počáteční data nejsou často ve stavu vhodném k přímému dolování dat. Procházejí proto následujícími kroky, které je připraví k potřebám dolovací úlohy. Samotná příprava je závislá na typu vstupních dat a dolovací úloze. V případě, že není některý krok přípravy dat zapotřebí, lze jej vynechat. Na kvalitě předzpracování dat z velké míry záleží úspěšnost dolovací úlohy.

2.3.1 Čištění dat

Během čištění dat dochází k hledání nezadaných hodnot, nekonzistencí v databázi, zašumění dat a následně pak dochází k vhodnému odstranění těchto problémů.

Nezadané hodnoty jsou nejčastějším problémem vyskytujícím se ve vstupních datech. Pro dolovací úlohy je vhodné připravit co největší množství odpovídajících dat, aby bylo možno

dosáhnout co nejlepších výsledků dolování. Proto se snažíme tyto hodnoty vhodně nahradit. Použít můžeme různé metody:

- Odstranění celého záznamu s chybějící hodnotou – tím zbytečně přicházíme o mnoho důležitých dat.
- Nahrazení uživatelem – v případě velkého objemu dat je tato možnost nepoužitelná.
- Náhrada globální konstantou – nevhodně zvolená konstanta může značně ovlivnit dolovací algoritmus.
- Náhrada průměrnou hodnotou ostatních hodnot téhož atributu – nejjednodušší metoda s relativně dobrým výsledkem.
- Náhrada průměrem stejných atributů prvků ve stejné třídě – pokud lze záznam klasifikovat, dojde k nahrazení průměrnou hodnotou prvků ve stejné třídě. Více je o klasifikaci uvedeno v kapitole 2.4. Tato metoda je lepší než celkový průměr.
- Náhrada nejpravděpodobnější hodnotou – získanou například pomocí predikce a klasifikace je nejkvalitnější nahrazení. Více o klasifikaci a predikci je uvedeno v kapitole 2.4.

Šum v datech vzniká náhodnými chybami při přenosu dat, chybami software, hardware nebo i chybami lidí. Zároveň se šumem lze ale v datech nalézt také odlehlé hodnoty, které však nesou skutečnou informaci a patří do dolovacích dat. Je velmi složité detekovat šum od odlehlých hodnot. Existují dva způsoby řešení tohoto problému:

1. Odlehlé hodnoty jsou důležité pro dolovací úlohu, pak je vhodné odfiltrovat z dat pouze hodnoty skutečně nesmyslné, jako například záporný plat zaměstnance.
2. Odlehlé hodnoty a šum v datech negativně ovlivňují dolovací úlohu, pak je možné je odstranit pomocí regrese nebo pomocí metod shlukování. Obě tyto operace jsou blíže popsány v kapitole 2.4 a ve výsledku vyhlazují šum a odlehlé hodnoty.

V případě, že je vhodné šum z dat odstranit, používají se následující techniky pro vyhlazení zašuměných dat:

- **Plnění** – seřazená posloupnost dat je rozdělena na přibližně stejné části zvané koše. Vyhlazení se následně provede třemi možnými způsoby: nahrazením všech dat v koši jejich průměrem, případně mediánem, nebo lze určit nejmenší hodnotu a největší hodnotu v koši a ostatní hodnoty nahradit nejbližší hodnotou z určených.
- **Regrese** – data lze vyhladit aproximací pomocí lineární regrese. Podrobné informace jsou uvedeny v kapitole 3.
- **Shlukování** – hodnoty, jež nenáleží žádnému shluku, můžeme označit jako šum a lze je tedy odstranit. Podrobnější popis shlukování se nachází v kapitole 2.4.4.

2.3.2 Integrace dat

K integraci dat dochází při slučování různých záznamů z různých zdrojů. Při této operaci se mohou vyskytnout následující problémy, které je třeba vyřešit před samotným dolováním:

- **Konflikt databázového schématu** – jedná se o konflikt v metadatech jednotlivých tabulek.
- **Konflikt hodnot** – vzniká používáním různých měr po světě, například délka může být uložena v mílích a kilometrech.
- **Konflikt identifikace** – stejně pojmenované sloupce tabulky mohou obsahovat různé informace.
- **Redundance** – ve sloučené databázi se některé stejné informace nacházejí na více místech.

2.3.3 Výběr dat

Vstupní dolovací data jsou často velmi rozsáhlá, proto je vhodné vybrat jen ty záznamy, které souvisí s dolovací úlohou. Tímto omezením lze docílit kratšího času zpracování dat dolovacím algoritmem a lze také značně ovlivnit celkový výsledek dolování. Můžeme vybírat pouze některé záznamy nebo jen některé atributy záznamů. Bohužel vhodný výběr je často součástí dolovacích experimentů a nelze jej často předem určit.

2.3.4 Transformace dat

Transformací dat rozumíme úpravu dat do vhodnějšího formátu pro dolovací metodu. Lze provést sumarizaci, agregaci, segregaci, normalizaci či redukci:

- **Sumarizace** - statistické matematické operace nad atributem záznamů, například součet významných záznamů
- **Agregace** - sloučení několika atributů záznamů do jediného, například (PSC,Město,Ulice) -> (Adresa)
- **Segregace** – rozdělení jednoho atributu záznamů na více atributů, například (Adresa) ->(PSC,Město,Ulice)
- **Normalizace** – převod rozsahu hodnot atributů do předem stanoveného intervalu, typicky <0,1> nebo <-1,1>
- **Redukce** – výběr jen některých atributů při zachování integrity a charakterů původních dat, například redukce maximální dimenzionality vstupních dat

2.4 Typy dolovacích úloh a algoritmy jejich řešení

Po komplexní přípravě dat popsané v minulé kapitole lze přejít k jádru procesu získávání znalostí z databází, a to ke specifikaci dolovací úlohy a výběru algoritmu k jejímu řešení. Výběr dolovací úlohy a příslušného algoritmu záleží na tom, jaký druh modelu dat se snažíme získat.

Základní dělení dolovacích úloh provádíme do dvou skupin:

- **Deskriptivní** – tyto úlohy charakterizují obecné vlastnosti zkoumaných dat. Výsledkem jsou tedy charakteristické vlastnosti analyzovaných dat.
- **Prediktivní** – tyto úlohy pomocí analýzy vstupních dat provádějí předpověď budoucích hodnot dat.

V následujících kapitolách budou popsány jednotlivé typy dolovacích úloh.

2.4.1 Popis konceptu/třídy

Koncept nebo třídu můžeme popsat dvěma způsoby. Pomocí charakterizace dat nebo diskriminace dat.

- **Charakterizace dat** – výsledným cílem je dosáhnout sumarizace obecných vlastností daného konceptu nebo třídy, tedy hledáme popis třídy z jejích vlastností.
- **Diskriminace dat** – výsledkem je také popis třídy, není však dosažen pomocí vlastností dané třídy, ale je vymezen ve vztahu s jinou nebo jinými rozdílovými třídami. Neboli vyhledáváme atributy a jejich hodnoty, ve kterých se nejvíce liší.

2.4.2 Frekventované vzory a asociační analýza

Vzory, které se v datech vyskytují často, nazýváme frekventované vzory a jsou základem pro asociační analýzu. Cílem dolovací úlohy je nalézt ve frekventovaných vzorech asociační pravidla. Nejtypičtějším příkladem dolování asociačních pravidel je analýza nákupního košíku.

V obchodě s potravinami je databáze tvořena informacemi o jednotlivých nákupech. V těchto informacích se velmi často vyskytují různé frekventované vzory, ze kterých můžeme zjistit, jaké kombinace zboží a s jakou pravděpodobností zákazníci nakupují. Dolováním z těchto dat můžeme získat například následující asociační pravidlo

$$koupí("chleba") \Rightarrow koupí("máslo")(podpora = 2\%, spolehlivost = 70\%),$$

kde podpora značí, v kolika procentech ze všech nákupů si zákazník koupil chleba i máslo, a spolehlivost vyjadřuje, u kolika procent z těchto nákupů došlo v případě zakoupení chleba zároveň i ke koupi másla. Zajímavost a použitelnost takových pravidel můžeme určit předem stanovenou minimální podporou a spolehlivostí.

Asociační pravidla mohou být jednodimenzionální, například první uvedené pravidlo obsahuje jednu dimenzi *koupí*, nebo vícedimenzionální, například následující pravidlo obsahuje navíc dimenzi pohlaví.

$$koupí("chleba") \wedge pohlaví("muž") \Rightarrow koupí("máslo")(podpora = 2\%, spolehlivost = 70\%)$$

Nejčastěji používané algoritmy pro dolování asociačních pravidel jsou algoritmus Apriori a FP-tree. Druhý jmenovaný je rychlejší, protože nemusí provádět generování kandidátů. Více o těchto algoritmech lze dohledat v [1].

Pro dolování multidimenzionálních asociačních pravidel se používají metody statistické diskretizace kvantitativních atributů, založené na vzdálenosti, využívající statistické analýzy nebo evoluční algoritmy. Podrobnější informace k jednotlivým metodám lze nalézt v [1].

Problém získaných asociačních pravidel vystupujících z dolování je jejich velké množství. Řešením je vytvoření různých omezení pro jejich výběr.

2.4.3 Klasifikace a predikce

Klasifikace a predikce jsou zástupci prediktivních dolovacích úloh. Tedy slouží k analýze existujících dat, která následně slouží k předpovídání zařazení hodnot nových. Je tedy nutné mít v počátku dostatek dat pro počáteční analýzu. Toho je často využíváno k automatickému rozhodování společností při příchodu nových dat o jejich zařazení do příslušné kategorie.

Nejčastěji bývá využití demonstrováno na bankovním sektoru. Z již uložených dat lze klasifikovat zákazníky jako schopné splácet úvěr a neschopné splácet úvěr v kombinaci různých faktorů. V případě nově příchozího zákazníka již tedy může být predikována jeho schopnost splácet úvěr předem a mohou mu být nabídnuty vhodné produkty, nebo v opačném případě mu jeho žádosti budou zamítnuty.

2.4.3.1 Klasifikace

Základním cílem klasifikace je rozdělení dat do různých tříd na základě jejich vlastností. Klasifikace je určena především pro predikci diskrétních hodnot. Proces klasifikace je rozdělen do tří fází:

1. **Trénovací fáze** – ze zadaných dat je vybrána trénovací množina. Prvky této množiny musí být klasifikovány do nějaké třídy, tedy jsme schopni určit správný výsledek klasifikace. Na této množině si klasifikační algoritmus vytvoří seznam pravidel, podle kterých se bude dále rozhodovat.

2. **Testovací fáze** – ze zbytku dat je vytvořena testovací množina. U této množiny také všechny prvky musí být klasifikovány, aby bylo možné rozhodnout, zda se danému algoritmu klasifikace zdařila. Tyto informace jsou však klasifikátoru úmyslně zatajeny a ten se snaží data z testovací množiny správně klasifikovat. Výsledkem této fáze je tedy úspěšnost klasifikačního algoritmu v procentech. Pokud je úspěšnost nedostačující, je zapotřebí se vrátit k trénovací fázi.
3. **Klasifikační fáze** – v této fázi jsou klasifikačnímu algoritmu předávána nová data a ten je klasifikuje do příslušných tříd. Jeho úspěšnost je dána kvalitou předchozích dvou fází.

Ke klasifikaci se využívají metody rozhodovacího stromu, neuronové sítě, jednoduché bayessovské klasifikace, bayessovské sítě, klasifikátory založené na k-nejbližším sousedovi, klasifikátory na fuzzy množinách a genetických algoritmech.

2.4.3.2 Predikce

Oproti klasifikaci je predikce vhodná pro předpovídání obecně spojitých veličin. Pro příklad je v literatuře často uváděna predikce platu zaměstnance v závislosti na zkušenostech, kvalifikaci a dalších vlastnostech. Proces predikce prochází stejnými fázemi jako klasifikace.

Nejčastěji používanou metodou pro predikci je regrese. V závislosti na vstupních datech a charakteristice očekávaných výsledků je nejčastěji používána lineární jednoduchá regrese, lineární vícenásobná regrese, popřípadě nelineární regrese.

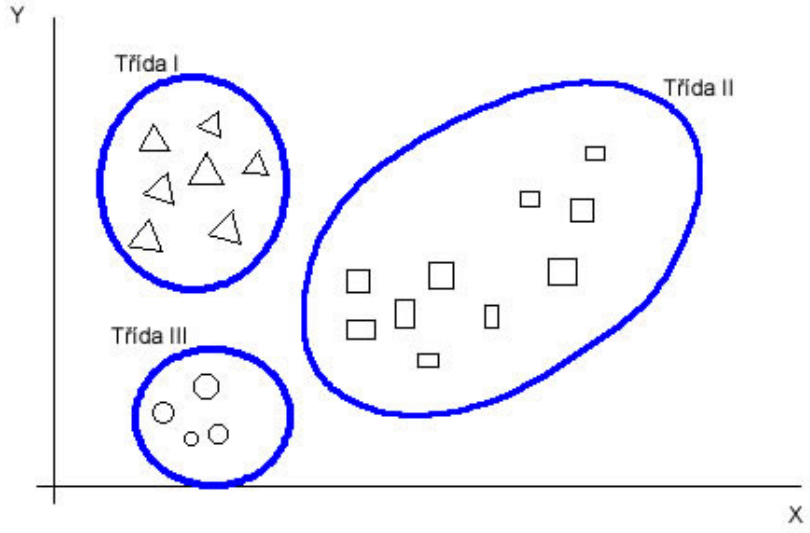
Vzhledem k zaměření celé práce je této dolovací metodě věnována samostatná kapitola 3.

2.4.4 Shluková analýza

Shluková analýza rozděluje data do tříd podle nějaké vlastnosti, ve které jsou si data nejvíc podobná. Tím vznikají shluky dat s co nejpodobnější vlastností a zároveň s co nejodlišnější vlastností od jiného shluku (obr. 2.2). Podobnost dat se často určuje podle vzdálenostní funkce jednotlivých atributů.

Shluková analýza je velice široký pojem. Je známo mnoho shlukovacích metod a jejich použití je závislé na konkrétním cíli dolování. Obecně můžeme dělit shlukovací metody následovně s uvedením příkladů algoritmů jejich řešení:

- Metody založené na rozdělování – algoritmy K-means, K-meloids
- Hierarchické metody – algoritmy AGNES, DIANA
- Metody založené na hustotě – algoritmy DBSACN, DENCLUE
- Metody založené na mřížce – algoritmus WaveCluster
- Metody založené na modelech – algoritmus Expectation – Maximization
- Metody pro shlukování vysoce dimenzionálních dat – algoritmus ClIQUE, PROCLUS



Obrázek 2.2 Shluková analýza

2.4.5 Dolování odlehlých hodnot

U většiny dolovacích metod odlehlé hodnoty často způsobují, že nedostáváme přesné informace. Proto je výhodnější je v části předzpracování odfiltrovat a tím zajistit větší přesnost dolovacích algoritmů. Ne vždy jsou však tyto hodnoty bezvýznamné. Často se v nich mohou nacházet zajímavé informace o nestandardním chování, například podvodné transakce na účtech zákazníků bank.

Pro získání takovýchto odlehlých hodnot používáme statistické metody, případně upravené dolovací metody. Například při využití shlukování odlehlé hodnoty nenáleží žádnému shluku nebo jsou klasifikovány do jedné třídy. Této problematice je věnována celá čtvrtá kapitola.

2.4.6 Evoluční analýza

Analyzuje rozprostření dat v čase. Jejich pravidelnost, intervaly změn a jejich trendy. Je tedy vhodná pro objekty, které se v čase mění. Využití této metody můžeme hledat například v dolování znalostí z burzovních transakcí a získané informace využít pro odhad dalšího vývoje burzy a rozhodování o investicích.

2.5 Vyhodnocení získaných výsledků

Vyhodnocení získaných výsledků je důležitou součástí dolovacího procesu. Je vždy zapotřebí vybrat pouze relevantní výsledky vhodné pro zadavatele dolovací úlohy. Řešení tohoto problému můžeme zčásti automatizovat zavedením metrik relevance získaných výsledků. Tento přístup se hodí

především tam, kde dolovací algoritmus vrací často podobné nebo nevýznamné vzory. Nebo můžeme relevanci získaných výsledků nechat na zkušenostech uživatele.

Součástí vyhodnocení získaných výsledků je také vhodná vizualizace například přehlednou tabulkou, různými typy grafů tak, aby i technicky nezkušený zadavatel či koncový uživatel získal přehled o výsledcích celého dolovacího procesu.

3 Predikce

Predikce je jednou z prediktivních dolovacích úloh, která slouží k přiřazování hodnot datům obecně spojitého charakteru, jak již bylo naznačeno v předchozí kapitole. Tato kapitola se věnuje této dolovací úloze podrobně.

Proces predikce stejně jako proces klasifikace prochází třemi fázemi:

1. **Trénovací fáze** – ze zadaných dat je vybrána trénovací množina. Množina by ideálně měla obsahovat dostatečný počet hodnot k popisu závislostí mezi atributy dat v celém jejich rozsahu. Na této množině proběhne takzvané trénování, neboli příprava a nastavení prediktivní metody ze známých hodnot atributů dat.
2. **Testovací fáze** – ze zbytku počátečních dat je vytvořena testovací množina, na které proběhne otestování nastavení prediktivní metody. Výsledkem této fáze je tedy úspěšnost klasifikačního algoritmu v procentech. Pokud je úspěšnost nedostačující, je zapotřebí se vrátit k trénovací fázi a vybrat jiná trénovací data, případně zvýšit jejich množství nebo zvolit jinou predikční metodu.
3. **Predikční fáze** – v této fázi již správně natrénovaná a ověřená predikční metoda vrací predikci nových dat v závislosti na požadavcích dolovací úlohy.

Z uvedeného popisu je zřejmé, že ke správné funkci predikční metody jsou zapotřebí dostatečná vstupní data. Ta je zapotřebí vhodně rozdělit na trénovací a testovací množinu. Důležitou podmínkou je, aby data v obou množinách byla navzájem nezávislá. Na toto dělení můžeme použít dvě základní metody:

- **Blokování** – data jsou náhodně rozdělena na trénovací a testovací skupinu nejčastěji v poměru 2:1.
- **Křížová kontrola** – data jsou také náhodně rozdělena do konečného počtu *k* množin *M*, kde každá množina M_1 … M_k obsahuje přibližně stejný počet prvků. Následně pak v jednotlivých *k* iteracích je v každé iteraci vybrána jedna množina pro testování a zbytek množin jsou použité jako trénovací.

Pro dolovací úlohy predikce se používá regresní analýza. Tu můžeme označit jako metodu, pomocí níž odhadujeme hodnotu náhodné veličiny (závisle proměnné, cílové proměnné, regresandu anebo vysvětlované proměnné) na základě znalosti jiných veličin (nezávisle proměnných, regresorů, kovariát anebo vysvětlujících proměnných) [3]. Tedy jinak řečeno hledáme funkci *F* s jednou nebo více nezávislých proměnných (x1,x2,…xn), množinu parametrů (a1,a2,..an) a s odchylkou *e*, která nejlépe kopíruje body ze zadané množiny dat

$$y = F(x, a) + e, \tag{3.1}$$

kde y neboli závislá proměnná je skalár nebo vektor z nějakého lineárního prostoru, odchylka e je rozdíl mezi známou cílovou hodnotou a předikovanou hodnotou. Při fázi učení se snažíme tuto odchylku co nejvíce minimalizovat. Její velikost pak následně určuje kvalitu regrese. Podle typu funkce F pak rozlišujeme různé typy regrese na lineární jednoduchou, vícenásobnou nebo nelineární jednoduchou a vícenásobnou.

3.1 Lineární jednoduchá regrese

Často chceme prozkoumat vztah mezi dvěma veličinami, kde jednu z nich bereme jako nezávisle proměnnou *x*, která má ovlivňovat závislou proměnnou *y*. K tomu nám slouží lineární regrese, která hledá aproximaci daných hodnot ve tvaru $(x_1, y_1),\ldots,(x_s, y_s)$ od $i=1,\ldots,s$ pomocí polynomu prvního řádu neboli přímkou o rovnici ve tvaru (3.2) tak, aby součet druhých mocnin odchylek jednotlivých bodů od přímky byl minimální.

$$Y = aX + b \tag{3.2}$$

Kde $x_1, x_2, \ldots, x_s$ – značí vstupní atributy daných dat,

$y_1, y_2, \ldots, y_s$ – značí výstupní atributy skutečných dat,

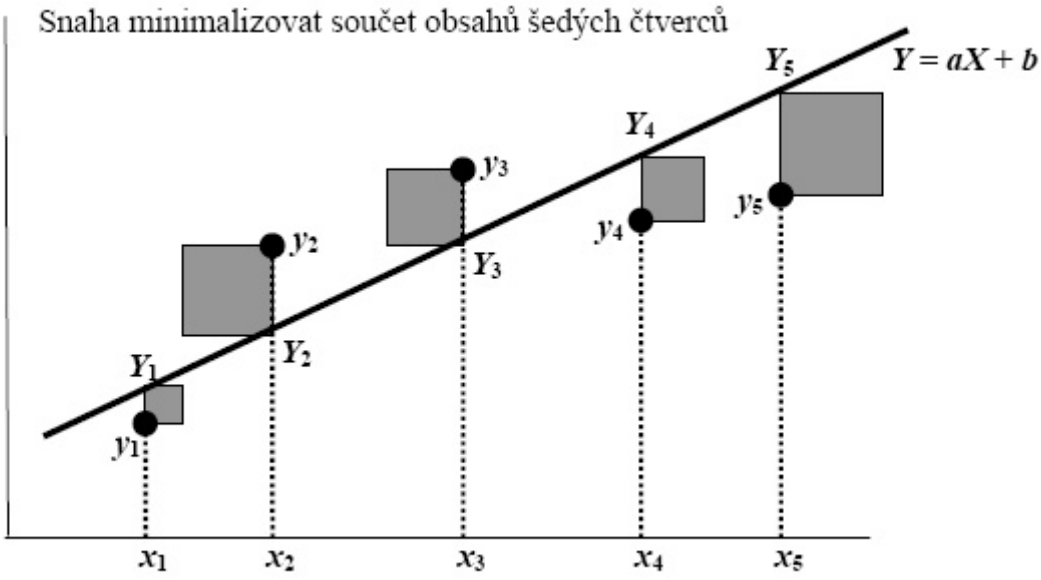
$Y_1, Y_2, \ldots, Y_s$ – jsou hodnoty, které byly získány dosazením vstupního parametru do dané rovnice přímky (tedy ty hodnoty, které budou pro dané vstupní atributy předpovězeny).

Nejčastěji používanou metodou pro výpočet koeficientů *a* a *b* je metoda nejmenších čtverců. Metoda nejmenších čtverců je založena na principu, že součet druhých mocnin rozdílu skutečné hodnoty y_i a aproximované hodnoty Y_i je minimální. Tato myšlenka je ilustrována na obrázku 3.1. Popis metody je převzat z [1].

Koeficienty *a*, *b* můžeme spočítat pomocí rovnice 3.3:

$$a = \frac{\sum_{i=1}^{s}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{s}(x_i - \bar{x})^2}; \qquad b = \bar{y} - a\bar{x}, \tag{3.3}$$

kde $\bar{x}$ značí aritmetický průměr hodnot $x_1, x_2, \ldots, x_s$ a y značí aritmetický průměr hodnot $y_1, y_2, \ldots, y_s$.



Obrázek 3.1 Ilustrace metody nejmenších čtverců - převzato z [1]

3.2 Lineární vícenásobná regrese

V některých případech je zapotřebí vzít k výpočtu predikce více nezávislých atributů vstupních dat, protože sledovat vztahy mezi dvěma proměnnými je často velké zjednodušení skutečnosti. Pro tyto případy je výhodnější použít lineární vícenásobnou regresi. Použitou metodu z předchozí kapitoly lze zobecnit na obecně *v* atributů [1]. Data tedy budeme očekávat ve tvaru (3.4):

$$(x_{11}, x_{12}, ..., x_{1v}, y_1), (x_{21}, x_{22}, ..., x_{2v}, y_2), ..., (x_{s1}, x_{s2}, ..., x_{sv}, y_s) \tag{3.4}$$

Aproximaci provedeme pomocí rovnice (3.5):

$$Y = a_0 + a_1 X_1 + a_2 X_2 + ... + a_v X_v \tag{3.5}$$

Je tedy potřeba najít hodnoty koeficientů a_0, a_1, a_2, …, a_v. Sestavme ze získaných dat matice *X* a *Y* následovně:

$$X = \begin{pmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1v} \\ 1 & x_{21} & x_{22} & \cdots & x_{2v} \\ 1 & \vdots & \ddots & & \vdots \\ \vdots & \vdots & & \ddots & \vdots \\ 1 & x_{s1} & x_{s2} & \cdots & x_{sv} \end{pmatrix} \qquad Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_s \end{pmatrix} \tag{3.6}$$

Výsledek hledejme jako matici *A*, která má tvar:

$$A = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_v \end{pmatrix} \tag{3.7}$$

Vzorec, podle kterého určíme výslednou matici *A* (a tím jednotlivé koeficienty), je:

$$A = (X^T X)^{-1} X^T Y\,, \tag{3.8}$$

kde horní index *T* značí matici transponovanou a horní index –1 matici inverzní. V následujícím příkladu je znázorněn praktický výpočet vícenásobné regrese pro dva nezávislé atributy X_1,X_2 a jeden závislý Y. Vstupní data jsou znázorněna v tabulce 3.1.

X_1	X_2	Y
0	0	2
1	1	2
-1	1	0

Tabulka 3.1 Vstupní data pro výpočet vícenásobné regrese

Budeme tedy hledat tuto regresní funkci:

$$Y = a_0 + a_1 X_1 + a_2 X_2 \tag{3.9}$$

Hodnoty z tabulky 3.1 můžeme přepsat do matic *X* a *Y* (3.10) a provést výpočet koeficientů a_0, a_1, a_2 dle výpočtu 3.11.

$$X = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & -1 & 1 \end{pmatrix} \qquad Y = \begin{pmatrix} 2 \\ 2 \\ 0 \end{pmatrix} \tag{3.10}$$

$$X^T X = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & -1 \\ 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & -1 & 1 \end{pmatrix} = \begin{pmatrix} 3 & 0 & 2 \\ 0 & 2 & 0 \\ 2 & 0 & 2 \end{pmatrix} \tag{3.11}$$

$$(X^T X)^{-1} = \frac{1}{4} \begin{pmatrix} 4 & 0 & -4 \\ 0 & 2 & 0 \\ -4 & 0 & 6 \end{pmatrix}$$

$$A = (X^T X)^{-1} X^T Y = \frac{1}{4} \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & -1 & 1 \end{pmatrix} \begin{pmatrix} 4 & 0 & -4 \\ 0 & 2 & 0 \\ -4 & 0 & 6 \end{pmatrix} \begin{pmatrix} 2 \\ 2 \\ 0 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \\ -1 \end{pmatrix}$$

Hledané koeficienty tedy jsou $a_0 = 2, a_1 = 1, a_2 = -1$ a výsledná regresní funkce je tvaru:

$$Y = 2 + X_1 - X_2 \tag{3.12}$$

Výsledná rovnice je rovnice plochy, kterou můžeme znázornit ve trojrozměrném grafu. V případě zpracování více nezávislých proměnných již však není možné graficky znázornit výsledky.

3.3 Nelineární regrese

V předchozích příkladech bylo předpokládáno, že aproximace přímkou nebo rovinou bude dostatečná. V popisech některých vztahů a jevů tento přístup nemusí být dostatečně popisující a predikce tak celkově selhává. V těchto případech může být vhodné použít nelineární regresi a pokusit se o lepší výsledky aproximace nějakou křivkou.

Řešení nelineární regrese nejčastěji provádíme transformací na lineární vícenásobnou regresi. Pro příklad můžeme zvolit aproximaci pomocí kubické funkce (3.13).

$$Y = a_0 + a_1 X + a_2 X^2 + a_3 X^3 \tag{3.13}$$

Lze definovat nové proměnné následným způsobem $X_1 = X, X_2 = X^2, X_3 = X^3$ a dosadíme-li je zpět do rovnice 3.13, získáme tak novou funkci tvaru vícenásobné regrese

$$Y = a_0 + a_1 X + a_2 X_2 + a_3 X_3, \tag{3.14}$$

kterou lze vyřešit způsobem uvedeným v kapitole 3.2.

3.4 Zobecněné lineární modely

Předchozí kapitoly se zabývaly klasickým regresním lineárním modelem. Obecně jsou však vlastnosti vstupních dat často odlišné od vstupních předpokladů klasického lineárního modelu. V těchto situacích odhady parametrů získané klasickou metodou nejmenších čtverců nedosahují optimálních výsledků. Proto je zapotřebí použít robustnější techniky predikce a to zobecněné lineární modely (anglicky Generalized linear models). Ty i při narušení základních podmínek klasických modelů zachovávají dobré statistické výsledky i pro případy, že rozložení náhodných chyb v modelu není normální a střední hodnota vysvětlované proměnné není identickou funkcí lineárního predikátoru [16].

3.4.1 Podmínky klasického lineárního modelu

Vyjděme opět ze základní rovnice pro klasický lineární model upravené do následujícího tvaru.

$$Y_i = \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_k x_{ik} + \varepsilon_i \tag{3.15}$$

Kde Y_i je i-tá hodnota vysvětlované veličiny (regresandu),

x_{ij} je i-tá hodnota vysvětlujících proměnných (regresorů),

β_j jsou koeficienty modelu,

ε_i jsou neznámé náhodné chyby,

pro $i = 1,\ldots,n$ a $j = 1,\ldots,k$ a n je počet pozorování.

Nebo alternativně lze použít zápis pomocí matic:

$$Y = X\beta + \varepsilon \tag{3.16}$$

Kde Y je vektor n vysvětlovaných hodnot,

X je matice hodnot vysvětlujících proměnných o rozměrech $n_x k$,

β je vektor koeficientů modelu,

ε je vektor náhodných chyb a

n je počet pozorování.

Pro takto definovaný klasický lineární model platí následující podmínky (převzato z [16], [17] a [18]):

1. $E(\varepsilon_i) = 0$ pro každé $i=1,2,\ldots,n$

Střední hodnota náhodné složky je nulová. Tato podmínka znamená, že náhodná složka nepůsobí systematickým způsobem na hodnoty vysvětlované proměnné Y.

2. $D(\varepsilon_i) = \sigma^2$ pro každé $i=1,2,\ldots,n$

Rozptyl náhodné složky je konstantní. Tato podmínka vyjadřuje, že variabilita náhodné složky nezávisí na hodnotách vysvětlujících proměnných a tudíž i podmíněná variabilita vysvětlované proměnné nezávisí na hodnotách vysvětlujících proměnných a je rovna neznámé kladné konstantě σ^2.

3. $Cov(\varepsilon_i \varepsilon_j)=0$ pro každé $i \neq j=1,2,\ldots,n$

Kovariance náhodné složky je nulová. Tedy hodnoty náhodné složky jsou nekorelované a z toho vyplývá i nekorelovanost různých dvojic pozorování vysvětlované proměnné Y. Pojem kovariance značí střední hodnotu součinu odchylek obou náhodných veličin X,Y od jejich středních hodnot.

4. X je nestochastická (nenáhodná) matice.

Znamená to tedy, že vysvětlující proměnné jsou nenáhodné.

5. Matice X má plnou hodnost, tedy $h(X)=k < n$ (n je počet pozorování).

Tato podmínka vyžaduje, aby mezi vysvětlujícími proměnnými nebyla funkční lineární závislost, tedy v matici X nesmí existovat lineárně závislé sloupce. Tato vlastnost je z hlediska dolování dat velmi důležitá. Značí, že nad vstupními daty musí proběhnout kontrola lineárních závislostí. Počet vysvětlujících proměnných nesmí být větší než počet pozorování.

6. ε_i mají normální rozdělení pro každé i=1,2,…,n.

Z této podmínky vyplývá normalita i pro vysvětlovanou proměnnou Y. Náhodný vektor Y má potom n-rozměrné normální rozdělení s vektorem středních hodnot $X\beta$ a tomu odpovídající kovarianční maticí.

7. Parametry β_j, j=1,2,…,k mohou nabývat libovolných hodnot.

Vektor β tedy může být libovolný.

3.4.2 Zobecněný lineární model

K zobecnění lineárního modelu dojde, pokud nahradíme podmínky uvedené v předchozí kapitole za obecnější. Pro různá vstupní data se používají odlišná zobecnění založená na zobecnění jedné podmínky z klasického lineárního modelu. Tato zobecnění se dají kombinovat a tím vytvářet různé zobecněné lineární modely. Následuje výčet několika obecně známých zobecnění:

- **Heteroskedascita** – připouští nekonstantní rozptyly náhodné veličiny, ale náhodné veličiny zůstávají nekorelované. Jedná se o zobecnění podmínky č. 3.
- **Korelované náhodné složky v modelu** – náhodné složky v modelu mohou být lineárně závislé. Zobecněná podmínka č. 3.
- **Regresory mohou být stochastické** – zobecnění podmínky č. 4. Vstupní data mohou obsahovat i náhodná data.
- **Náhodná složka může mít jiné než normální rozdělení** – zobecnění podmínky č. 6. Nejčastěji používané jsou normální, Poissonovo, gamma, inverzní Gaussovo rozložení podle vlastností vstupních dat.

Vždy je cílem zobecnit lineární model tak, aby lépe predikoval požadovanou veličinu na základě vlastností vstupních dat. Takový proces může probíhat postupně v závislosti na zkoumání vstupních dat a průběžných výsledků. Některá zobecnění nemusí k lepší predikci vést. Více podrobnějších informací lze nalézt v [16] a [17].

4 Dolování odlehlých hodnot

Obecně lze definovat odlehlou hodnotu jako hodnotu, která je netypická pro danou množinu dat. Tuto obecnou definici budeme dále v textu blíže specifikovat z různých hledisek získání odlehlých hodnot. Ne vždy totiž jsou nejhodnotnější informace obsaženy v obvyklých datech, která v jistém datovém souboru očekáváme. Získáním atypických hodnot ze vstupní datové množiny můžeme získat cenný přehled o anomáliích celého procesu, kterým jsou tato data vytvářena. Takovéto anomálie mohou například pocházet z nezákonných aktivit nebo z chyb v analyzovaném procesu. Příklad odlehlé hodnoty můžeme demonstrovat na bankovním účtu. Z dlouhodobého hlediska jsou transakce na tomto účtu nejčastěji do 10 000 Kč a cílové banky všech transakcí jsou pouze národní. Jako odlehlou hodnotu pak můžeme uvažovat v množině všech transakcí převod 10 000 000 Kč během krátké doby z banky ve Švýcarsku do banky na Bahamských ostrovech přes tento účet.

Téma odlehlých hodnot (anglicky outliers) se vyskytuje v procesu získávání znalostí z databází na dvou místech. Jednak v části přípravy a předzpracování dat a následně jako samostatná dolovací úloha.

4.1 Příprava a předzpracování dat

Jak již bylo zmíněno dříve v kapitole 2.3, může být součástí přípravy dat analýza odlehlých hodnot v závislosti na dolovací metodě a na citlivosti dolovacího algoritmu na tato data. Obecně tedy mohou nastat dva případy využití dolování odlehlých hodnot.

Algoritmus dolovací úlohy je citlivý na odlehlé hodnoty i šum a je tedy žádoucí je ze vstupních dat odstranit. K tomuto kroku je však potřeba tato data identifikovat. K tomu může být využita některá metoda dolování odlehlých hodnot z kapitoly 4.2.

V případě zadání dolovací úlohy typu dolování odlehlých hodnot je velice žádoucí tyto hodnoty v analyzovaných datech ponechat. Je však důležité z dat odstranit šum, který se v datech může nacházet vlivem různých chyb, které nemají s dolovací úlohou nic společného, a tedy by její výsledky mohl negativně ovlivnit. Během odstraňování šumu však nesmí dojít i k odstranění důležitých odlehlých hodnot. Rozpoznání šumu a skutečných významných odlehlých hodnot může být veliký problém.

Máme-li například databázi pacientů s údaji o jejich tělesné teplotě, můžeme šum z dat odfiltrovat jistými omezeními. Tělesná teplota pacienta by měla být v rozmezí 36°C – 42°C, tedy ostatní hodnoty mimo tento rozsah můžeme odstranit, případně nahradit nějakou opravnou metodou uvedenou v 2.3.1. Pokud však vznikne chyba v automatickém měřicím systému teploty pacientů a výsledná teplota padne u zdravého člověka do přijatelného rozsahu například na hodnotu 41,9°C, není

již jednoduché tento šum odstranit. Na řadu musí přijít odborník v daném oboru a hodnotu vyřadit dle svých zkušeností nebo lze použít nějakou prediktivní metodu pro úpravy těchto dat.

4.2 Metody dolování odlehlých hodnot

Cílem metod dolování odlehlých hodnot je jejich identifikace v množině vstupních dat. K jejich vyhledání můžeme využít dva úhly pohledu na řešenou problematiku. Matematický aparát statistických měr, který nemusí být vždy ideální a přizpůsobivý. Nebo můžeme využít jiných dolovacích metod přizpůsobených pro vyhledávání odlehlých hodnot.

4.2.1 Z pohledu statistických měr

Množina dolovacích dat je z pohledu statistiky soubor hodnot náhodných veličin, tedy jejich hodnotu nelze určit předem. Pro vyhledávání odlehlých hodnot je potřeba tuto množinu statisticky popsat různými statistickými mírami.

Nejčastěji používané míry jsou míry středu dat.

- **Průměr** – nejčastěji používaný jednoduchý popis středu dat.

$$\overline{X} = \frac{\sum_{i=1}^{N} x_i}{N} \tag{4.1}$$

- **Modus** – hodnota je hodnotou nejčastěji se vyskytujícího prvku. Může existovat více prvků se stejnou nejčastější hodnotou nebo se žádný prvek nevyskytuje vícekrát, pak modus neexistuje.
- **Medián** – hodnota mediánu odpovídá hodnotě, která se v seřazené posloupnosti dat nachází uprostřed. V případě sudého počtu se medián rovná průměru dvou prostředních prvků.

Další užitečnou mírou pro popis dat je míra variace. Značí rozptýlení hodnot kolem průměru.

- **Rozptyl**

$$\sigma^2 = \frac{1}{N}\sum_{1}^{N}(x_i - \bar{x})^2 \tag{4.2}$$

- **Směrodatná odchylka**

$$\sigma = \sqrt{\sigma^2} \tag{4.3}$$

- **Mezikvartilová vzdálenost** – kvantity jsou body z hranic pravidelných intervalů kumulativní distribuční funkce náhodné veličiny. Pokud seřazenou posloupnost hodnot

rozdělíme na čtyři intervaly, vznikají nám tím kvartily. S jejich pomocí můžeme definovat mezikvartilovou vzdálenost jako IRQ.

$$IRQ = Q3 - Q1 \tag{4.4}$$

Jako další míry lze použít míry šikmosti a špičatosti.

4.2.1.1 Pravidlo čtyř sigma

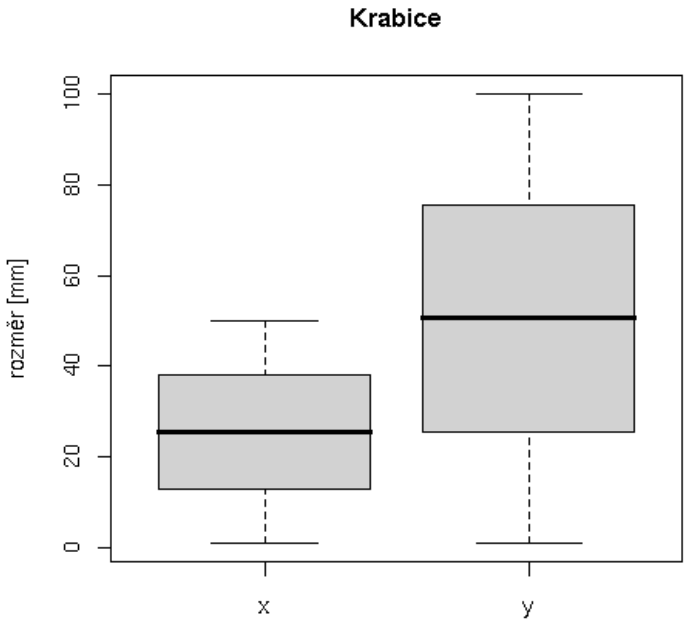
Pokud statistický soubor obsahuje alespoň 10 hodnot, lze pro detekci odlehlých hodnot použít pravidlo čtyř sigma [5], které se opírá o Bienaymé-Čebyševovu respektive Gaussovu nerovnost. Z těchto fundamentálních nerovností teorie pravděpodobnosti vyplývá, že v intervalu $(\bar{x} - 4s, \bar{x} + 4s)$, kde $\bar{x}$ respektive s jsou výběrový aritmetický průměr respektive výběrová směrodatná odchylka, leží 99,99 % hodnot pro výběry z normálního rozdělení, 97 % hodnot pro výběry ze symetrického unimodálního rozdělení a 94 % hodnot u výběrů ze zcela libovolného pravděpodobnostního rozdělení. Jestliže tedy některá hodnota bude ležet mimo výše zmíněný interval, lze s dostatečně vysokou pravděpodobností říci, že se jedná o odlehlou hodnotu.

4.2.1.2 Mezikvartilová vzdálenost

Je nejčastěji užívaná metoda pro detekci odlehlých hodnot s jednoduchou možností vizualizace výsledků pomocí krabicového grafu. Z určené mezikvartilové vzdálenosti IRQ pomocí vzorce 4.4 lze definovat jako odlehlé hodnoty všechny hodnoty nacházející se mimo rozsah intervalu

$$\langle Q1 - 1{,}5IRQ, Q3 + 1{,}5IRQ \rangle . \tag{4.5}$$

V krabicovém grafu (obr. 4.1) pak střední čárka v krabici představuje medián. Hranice krabice pak představují 1. a 3. kvartil. Oblast mezi 1. a 3. kvartilem se označuje jako interkvartilový interval (IQR). Odlehlé hodnoty (1,5 × IQR) představují koncové úsečky. Body, které se nacházejí ve větší vzdálenosti než 1,5 × IQR od Q1 respektive od Q3, jsou zobrazeny v podobě koleček [6] a označují nalezené odlehlé hodnoty.



Obrázek 4.1 Ukázka krabicového grafu - převzato z [6]

4.2.1.3 Další metody

Mezi další metody hledání odlehlých hodnot lze vyjmenovat metody založené na testovaném kritériu, například Grubsův test [7]. Všechny dosud uvedené metody pracují s jednorozměrnými daty. V případě vícerozměrných datových souborů je zapotřebí v některých případech použít specializované metody pro hledání odlehlých vektorů, například využívajících Mahalanobisovy vzdálenosti.

Definici odlehlých hodnot lze tedy konkretizovat z pohledu statistických měr jako následující hodnoty:

- jejichž absolutní hodnota se liší od průměru o více než zvolený násobek (7, 5, 4, 3) směrodatné odchylky (platí pro rozdělení blízké normálním) [8];
- které patří do prvního nebo posledního decilu všech hodnot;
- které patří mimo rozsah 1,5 × IQR;
- které jsou vyšší nebo nižší než předem stanovené meze pomocí stanoveného testu.

4.2.2 Z pohledu dolovacích úloh

Pro hledání odlehlých hodnot lze také využít různé metody ostatních dolovacích úloh. Po vhodné úpravě je lze přizpůsobit k detekci nebo predikci odlehlých hodnot. Definici odlehlých hodnot z pohledu dolovacích metod tedy můžeme shrnout tak, že se jedná o hodnoty, které jsou označené konkrétní dolovací metodou jako odlehlé.

4.2.2.1 Shluková analýza

Jak již bylo dříve v kapitole 2.4.4 naznačeno, shluková analýza rozděluje množinu dat do tříd podle jejich vlastností na základě podobnosti. Nejčastěji se využívá pro posouzení podobnosti objektů vzdálenostní funkce. Tímto lze pomocí shlukové analýzy dobře popisovat rozložení dat, čehož se dá využít při dolování odlehlých hodnot.

Výhodou shlukové analýzy oproti klasifikaci je, že nepotřebujeme žádná trénovací data ani předdefinované třídy. Jedná se tedy o případ učení bez učitele, kdy shlukování probíhá přímo na vstupní množině dat.

Obecně pro shlukovou analýzu platí, že je potřeba vybrat takové vlastnosti dat, které nejvíce rozlišují jednotlivé skupiny objektů od sebe. Pro dolování odlehlých hodnot je proto důležité vybrat takové vlastnosti, které dobře popisují vztahy mezi daty odpovídající dané úloze dolování odlehlých hodnot.

Výběr shlukové dolovací metody má zásadní vliv na výsledky dolování. Možnosti a přístupy k dolování odlehlých metod si ukážeme na různých metodách shlukové analýzy.

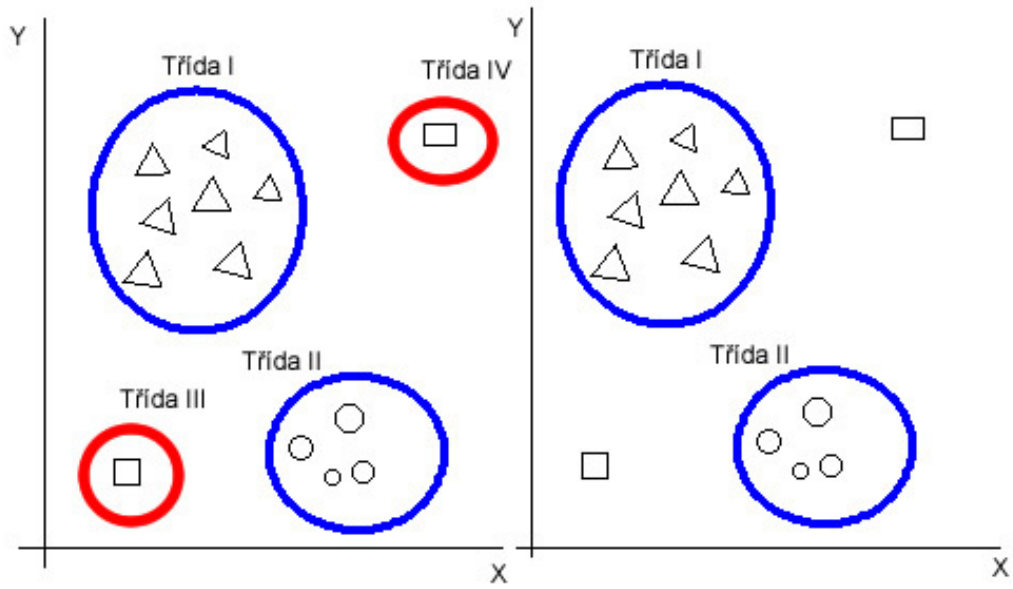
Metody založené na rozdělování potřebují v počátku určit počet tříd, do kterých se budou data rozdělovat. Jednotlivé třídy musí splňovat podmínky, že každá třída musí mít alespoň jeden prvek a každý prvek musí náležet pouze do jedné třídy. V prvním kroku shlukování jsou náhodně vybrány prvky ze vstupní množiny dat, které odpovídají jednotlivým třídám. Následně iterativně podle posuzování podobnosti jsou další prvky rozdělovány a přeskupovány v jednotlivých třídách. Toto přerozdělování je určeno metodou, například k-means nebo k-meloid. Výsledkem jsou tedy shluky dat v předem daném počtu tříd. Toto rozdělování je velmi závislé na počtu tříd, charakteru dat a porovnávaných vlastnostech. Odlehlé hodnoty se v tomto případě ideálně nacházejí v jedné třídě nebo naopak ve třídách o nejmenším počtu prvků. V této fázi je nejdůležitější prozkoumání výsledků a určení, zda vede k detekci odlehlých hodnot. V případě špatných výsledků je zapotřebí experimentálně vyzkoušet jiné počáteční parametry nebo vlastnosti k porovnání dat.

Hierarchické metody vytvářejí hierarchický rozklad vstupní množiny dat pomocí dvou přístupů. Metoda zdola-nahoru v první fázi umístí každý objekt do jedné třídy a dochází ke slučování nejpodobnějších tříd. Příkladem může být metoda AGNES. Metoda shora-dolů nejprve umístí všechny objekty do jedné třídy a tu následně rozděluje na menší dle podobnosti jednotlivých objektů. Příkladem může být metoda DIANA. Výsledkem této metody je strom shluků. Jeho podrobnost je dána nastavením konečného počtu tříd a zvolenou metrikou pro rozdělení nebo shlukování tříd. Z hlediska dolování odlehlých hodnot se opět snažíme tyto hodnoty nalézt v jedné třídě v případě, že jsou si navzájem podobné, nebo ve třídách s nejmenším počtem prvků. Přesnost dolování můžeme řídit konečným počtem tříd.

Metody založené na hustotě hledají shluky pomocí hustoty dat ve vstupní množině dat. Hlavním cílem je tedy identifikovat shluky dat s největší hustotou. Odlehlé hodnoty je však nutno hledat v těch nejméně významných shlucích. Případně je také můžeme nalézt jako hodnoty nenáležící

žádnému shluku a to v případě, že jejich výskyt je tak řídký, že nesplňuje minimální podmínku hustoty pro přiřazení k nějakému shluku. Jako příklady metod můžeme uvést DBSCAN a DENCLUE. Metody založené na hustotě vykazují dobré schopnosti vyhledání a identifikace odlehlých hodnot.

Pro úplnost lze uvést ještě další metody založené na mřížce, na modelech a speciální metody pro vysoce-dimenzionální data. Z uvedených příkladů je zřejmé, že zajímavé informace o odlehlých hodnotách se nacházejí v primárně nejméně důležitých třídách vytvořených shlukovacími metodami, popřípadě jsou to hodnoty, které nepatří žádným shlukům.



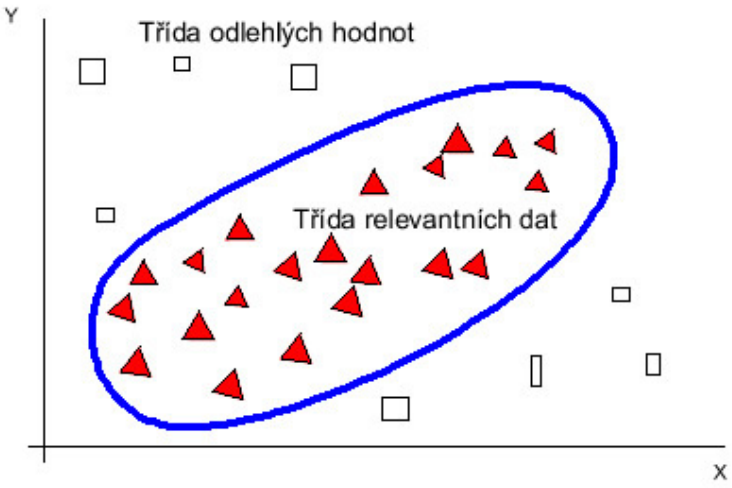
Obrázek 4.2 Odlehlé hodnoty ve shlukové analýze. Vlevo jsou odlehlé hodnoty klasifikovány do tříd III a IV. Vpravo nejsou klasifikovány do žádné třídy.

4.2.2.2 Klasifikace a predikce

Pojem klasifikace a predikce byl vysvětlen v kapitole 2.4.3. Jedná se o metody, které potřebují mít na začátku dostatečné množství relevantních dat a na jejich základě se naučí klasifikovat a predikovat nové hodnoty, tedy o metody s učitelem. Pro tyto metody je nejdůležitější trénovací množina dat. Tu je třeba vhodně upravit.

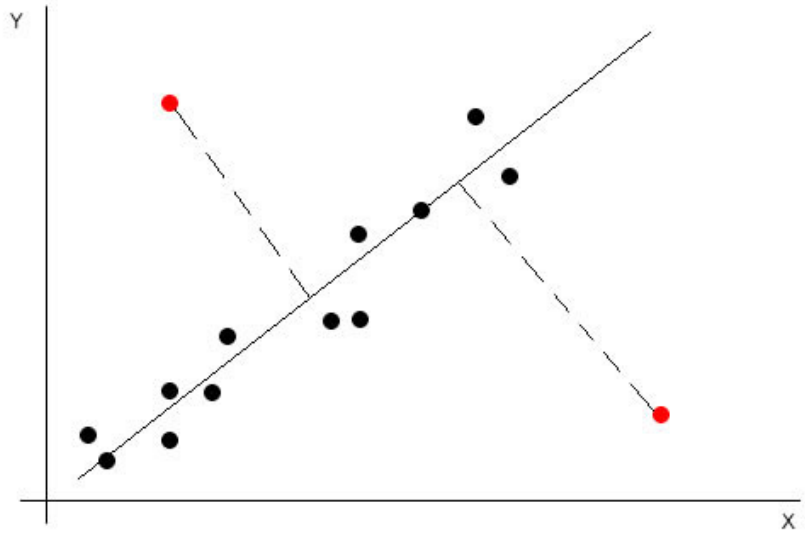
V případě klasifikace se zaměříme na dolování diskrétních odlehlých hodnot. Jako trénovací množinu zvolíme množinu všech dostupných dat rozdělenou na dvě podmnožiny. První označíme jako relevantní data, například hodnotou 0, a druhou část označíme jako odlehlé hodnoty hodnotou 1. Přítomnost množiny odlehlých hodnot nám pak následně dává možnost i otestovat klasifikátor. Toto rozdělení však může být velmi náročné a ve většině případů je nelze provést automatizovaně. Je tedy vhodnější a častěji se používá možnost pro trénovací fázi použít množinu pouze relevantních dat,

která bývá častěji dostupná. V tomto případě ale není možné provést testovací fázi, která se přeskočí. Testování je pak součástí aplikační fáze na nových datech. Výsledným cílem klasifikace tedy v obou případech bude klasifikace dat do dvou tříd - třída relevantních dat a třída odlehlých hodnot znázorněné na obrázku 4.3. Úspěšnost této metody v detekci odlehlých hodnot je opět ve kvalitě trénovací a testovací množiny a také ve zvolené klasifikační metodě. Pro klasifikaci je možné použít různé typy klasifikátorů, například založených na rozhodovacím stromu, neuronových sítích a bayessovské naivní klasifikaci.



Obrázek 4.3 Klasifikace a odlehlé hodnoty

V případě predikce se zaměříme na dolování spojitých odlehlých hodnot. Zde je zapotřebí použít jako trénovací množinu všechny dostupné relevantní hodnoty. Tím dojde k upravení regresních koeficientů do ideálního tvaru. Postup je popsán v kapitole 3. Nově příchozí hodnoty jsou pak detekovány jako odlehlé podle vzdálenosti od predikované hodnoty. Pokud překročí předem stanovenou mez, jsou označeny za odlehlé. Meze mohou být stanoveny na základě statistických údajů, získaných z trénovací množiny, nebo nastaveny v závislosti na potřebách dolovacího algoritmu.

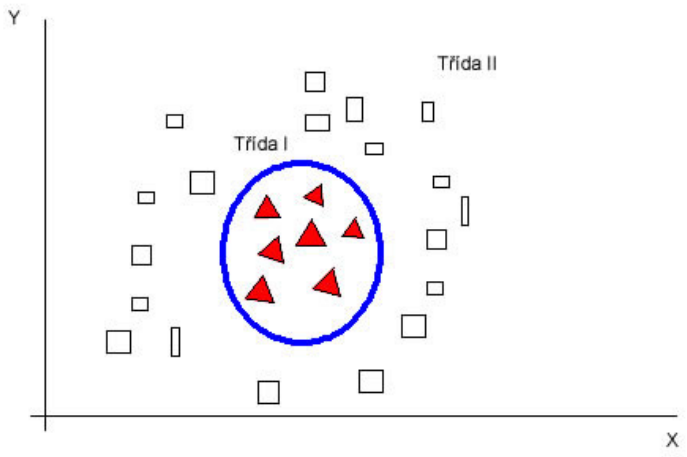


Obrázek 4.4 Jednoduchá lineární regrese a odlehlé hodnoty

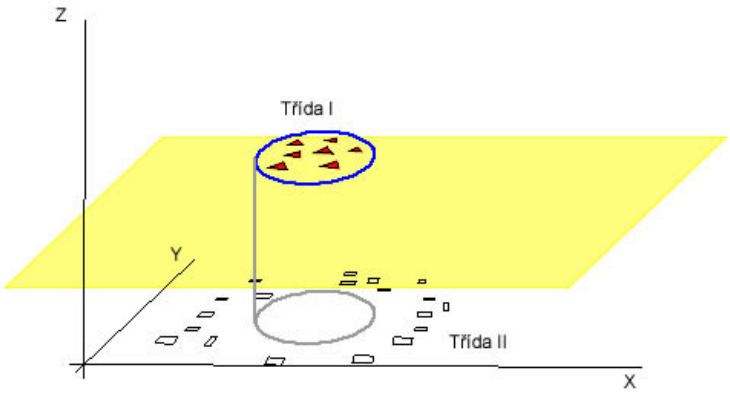
Jako příklad algoritmu řešícího dolování odlehlých hodnot na základě predikce a klasifikace můžeme využít v následující kapitole popsaný algoritmus Support vector machines, který v základu rozděluje vstupní množinu dat na dvě třídy. První bude obsahovat relevantní data a druhá odlehlé hodnoty.

4.3 Support vector machines – SVM

Support vector machines je v poslední době stále častěji používaný algoritmus pro klasifikaci i predikci. Jeho výhodou je schopnost práce s velkým množstvím dat, neomezenost počtem atributů a kvalitní schopnost učení na rozsáhlé trénovací množině.



Obrázek 4.5 Vstupní množina dat pro algoritmus SVM



Obrázek 4.6 Rozdělení prostoru podle algoritmu SVM

V základu algoritmus klasifikuje pouze do dvou tříd tak, že se snaží o nalezení lineární hranice pro rozdělení prostoru na dva podprostory s cílem oddělení dvou tříd na největší možnou vzdálenost. Jedním ze základních principů jak toho dosáhnout je převod daného vstupního prostoru do jiného vícedimenzionálního prostoru, kde se již i jinak nelineárně rozdělitelné třídy dají rozdělit lineárně. Nově vznikající prostory nejsou limitovány počtem dimenzí.

Způsob převodu dat ze vstupního prostoru na vícedimenzionální prostor řeší jádrová (kernel) funkce. Její vlastnosti určují výsledné vlastnosti SVM klasifikátoru a z tohoto důvodu algoritmus SVM řadíme mezi kernel-based algoritmy. Obecně lze tyto jádrové funkce rozlišovat na lineární a nelineární.

Tato schopnost je ideální pro využití k detekci odlehlých hodnot. Algoritmus je schopen pro vytvoření klasifikačního modelu použít i jen jednu vstupní množinu, kterou považuje za jednu třídu relevantních dat. V případě následného použití modelu pro detekci odlehlých hodnot jsou všechny hodnoty mimo tuto třídu klasifikovány jako odlehlé. Dalšími úpravami tohoto algoritmu jsme schopni řešit i úlohy regrese.

5 Dataminer

Od roku 2006 je na Fakultě informačních technologií VUT v Brně vyvíjen univerzální dolovací systém Dataminer. V systému jsou implementovány všechny kroky dolovacího procesu a tím je umožněno jeho široké využití v různých dolovacích úlohách. Jednotlivé dolovací metody jsou implementovány jako nezávislé moduly, jejich počet a různorodost zaměření je každým rokem zvětšována.

Dataminer je dvouvrstvá aplikace klient – server, kde klienta tvoří aplikace v jazyce Java vytvořená na platformě NetBeans a serverovou část tvoří databázový server od společnosti Oracle s podporou dolovacích metod Oracle data mining.

Klientská část aplikace je postavena především na jazyce DMSL [9], ve kterém jsou definována metadata, operace nad daty v procesu dolování a slouží ke komunikaci mezi jádrem aplikace a jednotlivými dolovacími moduly.

5.1 Platforma NetBeans

NetBeans je open source projekt vyvíjený firmou Sun Microsystems [10]. V rámci tohoto projektu existují dva hlavní produkty - vývojové prostředí NetBeans IDE a vývojová platforma NetBeans Platform.

Vývojové prostředí NetBeans IDE je nástroj primárně sloužící pro vývoj aplikací v jazyce Java, ale lze v něm vyvíjet aplikace i v jiných podporovaných jazycích.

Vývojová platforma NetBeans Platform je modulární a rozšiřitelný základ pro vytváření složitých a rozsáhlých aplikací, proto je vhodná pro vývoj Datamineru, který využívá především modularity.

5.2 Oracle Data Mining

Dolovací nástroj Dataminer používá pro uložení a práci s daty databázový systém od firmy Oracle, který obsahuje i podporu pro dolování dat. Lze tedy vyvíjet dolovací moduly, které využívají databázi pouze pro uložení dat a dolovací úloha probíhá na straně klienta, nebo lze využít podpory dolování dat Oracle Data Mining - ODM na straně databázového serveru.

Výhodou prvního řešení je možnost implementace vlastních algoritmů, nevýhodou může být složitější a méně efektivní práce s databází ze strany klienta. Jednoznačnou výhodou druhého řešení je využití již předem připravených dolovacích funkcí na úrovni jádra databázového serveru. Nevýhodou pak může být omezení na podporované algoritmy a nemožnost jejich detailního nastavení.

V současné době existuje nejnovější verze Oracle Database 11g Release 2, která rozšiřuje možnosti dolování pomocí nových algoritmů v ODM. V součtu tedy obsahuje následující dolovací metody:

- **Klasifikace** – pro klasifikaci jsou k dispozici naivní bayessovský klasifikátor, rozhodovací strom, SVM – support vector machines a nově přidaná logistická regrese založená na GLM – generalized linear models a s podporou dolování v textu a transakčních datech.
- **Regrese** – pro regresi jsou připravena řešení jak pro jednoduchou lineární regresi, tak pro vícenásobnou lineární regresi, využívající algoritmů SVM a GLM.
- **Dolování nejdůležitějších atributů** – slouží k hledání nejdůležitějších atributů pro využití v dalších dolovacích úlohách a využívá algoritmu minimální vzdálenosti.
- **Detekce odlehlých hodnot** – využívá metody klasifikace pomocí one-class SVM, kdy relevantní hodnoty klasifikuje do jedné třídy a jako odlehlé hodnoty detekuje ty, které tam nepatří.
- **Shlukování** – využívá algoritmů k-means a ortogonal partition clustering.
- **Asociační pravidla** – pro dolování asociačních pravidel je připraven algoritmus Apriori.
- **Extrakce nových atributů** – má za cíl z existujících atributů produkovat nové jako jejich lineární kombinace a k tomu využívá algoritmus non-negative matrix factorization.

Součástí ODM jsou také funkce pro předzpracování dat, jako transformace, vzorkování, plnění a diskretizace.

Propojení dolovacích funkcí a aplikace je umožněno pomocí jazyka PL/SQL nebo pomocí Java API standardu Oracle Data Mining Java API - OJDM implementujícím Java data mining – JDM verze 2.0 pod označením JSR 247 nebo starší verze 1.0 pod označením JSR 73. OJDM definuje rozhraní popisující propojení dolovacích funkcí s nezávislými dolovacími nástroji.

K rozhraní Java data mining pro účely propojení s ODM existuje přehledná dokumentace od firmy Oracle přístupná na [11]. K jednotlivým dolovacím metodám existují jednoduché ukázkové programy jejich použití v jazyce Java [21], ze kterých je doporučené vycházet při implementaci modulů.

5.3 DMSL

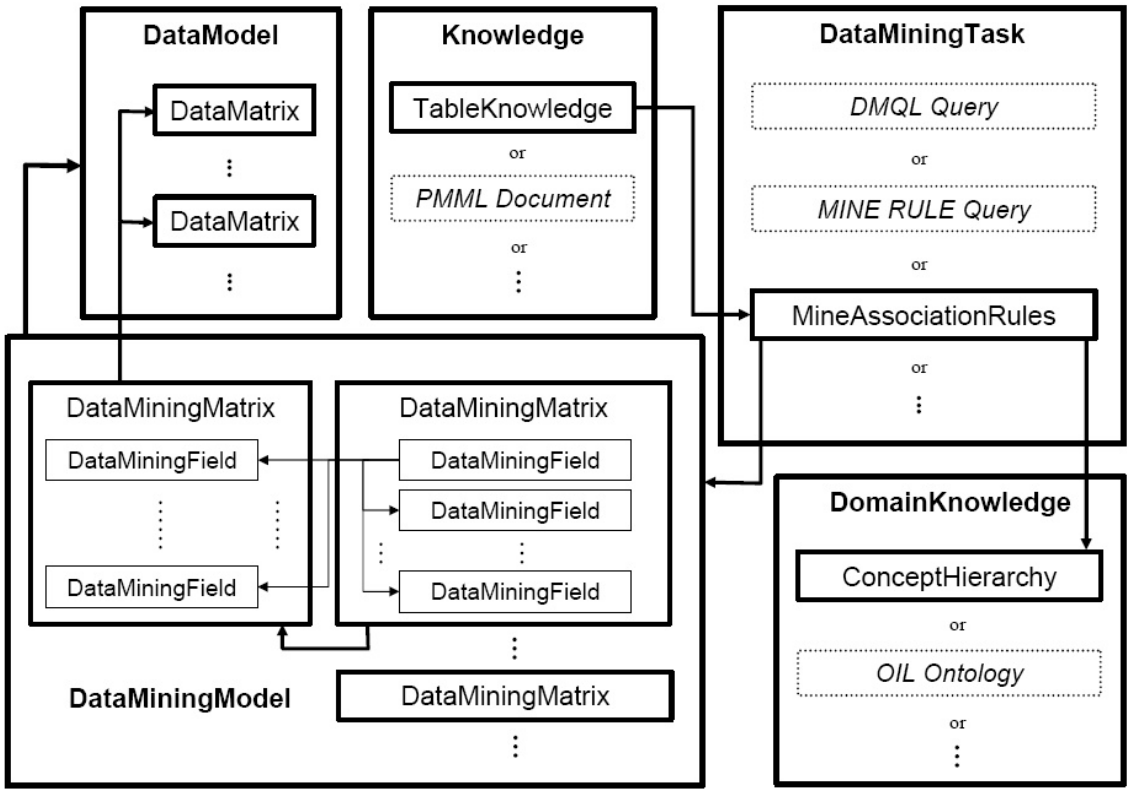
Jazyk DMSL je založený na struktuře jazyka XML a slouží k ukládání metadat a průběhu dolovacího procesu. Pomocí tohoto jazyka lze popsat celý dolovací proces od popisu vstupních dat přes

transformace až po popis samotné dolovací úlohy a výsledků dolování. Jazyk je neustále upravován pro potřeby jednotlivých dolovacích modulů.

Základní strukturu dokumentu tvoří následující elementy:

- Data Model – element sloužící k definici struktury platformě nezávislých vstupních dat.
- Data Mining Model – element slouží k popisu dat, která budou zahrnuta do dolování dat. Každý Data Mining Model se odkazuje na jeden DataModel.
- Domain Knowledge – element definuje informace o vztazích, integritních omezeních a oborech hodnot dat.
- Data Mining Task – element specifikující dolovací algoritmy a jejich parametry. Z hlediska různorodosti dolovacích úloh je jeho syntaxe volná a přizpůsobuje se potřebám dané dolovací úlohy.
- Knowledge – element zejména popisuje vydolované znalosti, ale může obsahovat i obecné informace z průběhu dolování. Tento element má také volnou syntaxi pro přizpůsobení výstupům různých dolovacích úloh.

Struktura a závislosti DMSL elementů jsou přehledně zobrazeny na obrázku 5.1. Více informací o DMSL lze získat v disertační práci Petra Kotáska [9].



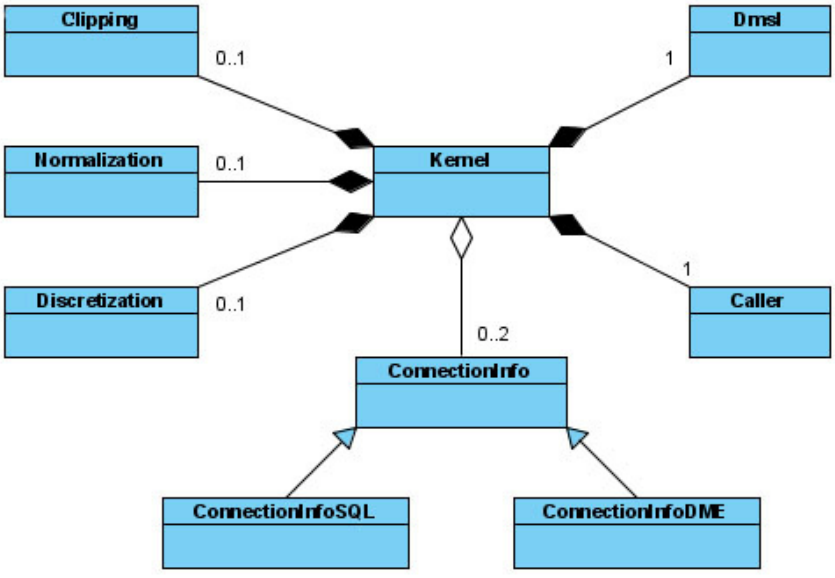
Obrázek 5.1 Struktura a závislosti DMSL elementů - převzato z [9]

5.4 Základní struktura systému

Systém můžeme rozdělit na jádro, grafické uživatelské rozhraní a nezávislé dolovací moduly. Tyto části vytvářejí komplexní řešení umožňující vytváření, ukládání a práci s projekty zastřešující celý dolovací proces. V jednotlivých projektech je umožněno načítání dat, jejich transformace a předzpracování, volba dolovacích úloh nad těmito daty a možnosti zobrazení výsledků jednotlivých fází dolovacího procesu. Stav celého systému a dolovacích úloh je uchováván v datové struktuře DMSL (kapitola 5.3).

5.4.1 Jádro systému

Základ jádra tvoří třída Kernel. Tato třída obstarává práci s projekty. Další třídy jsou navázány na tuto základní třídu a rozšiřují její funkčnost. Třída Dmsl zajišťuje práci s datovými strukturami uchovávajícími stav systému, projektů a dolovacích úloh. Datové struktury vycházejí ze struktury dokumentu DMSL. Důležitou funkcí jádra je správa připojení k databázi. Vytvářejí se dvě spojení - jedno pro práci s SQL dotazy a druhé pro práci s Data Mining Engine - DME. Jejich obsluhu mají na starosti třídy ConnectionInfoSQL a ConnectionInfoDME. Třída Caller slouží k volání externích tříd pro zpracování dat pomocí VIMEO funkcí (Valid, Invalid, Missing, Empty, Outlier). Dále jsou součástí jádra systému třídy pro předzpracování a transformaci dat. Struktura jádra je zobrazena na obrázku 5.2.



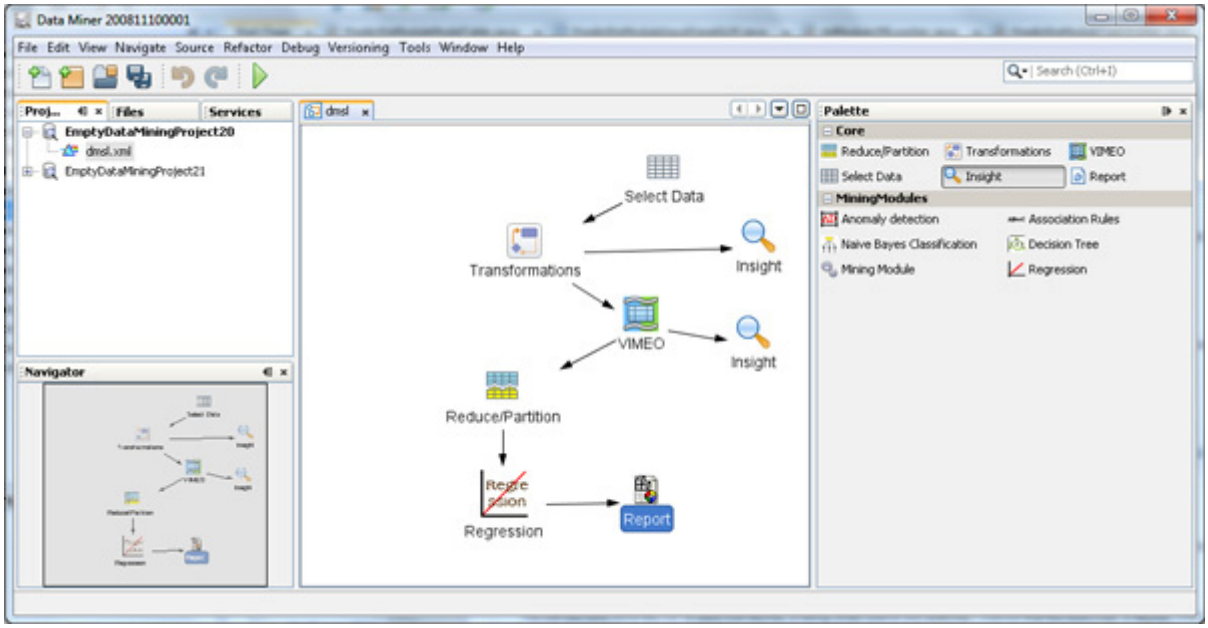
Obrázek 5.2 Struktura jádra, diagram tříd - převzato z [15]

5.4.2 Grafické uživatelské rozhraní

Grafické uživatelské rozhraní - GUI je nadstavbou nad jádrem systému. Je navrženo tak, aby umožňovalo práci s projekty a v rámci jednotlivých projektů modelování dolovacího procesu pomocí komponent (obr. 5.3). Komponenty se dají propojovat šipkami pomocí "ctrl + levý klik myši".

Funkčnost komponent je přístupná až po propojení do dolovacího procesu. Následuje výčet komponent:

- Select data – komponenta slouží k definici vstupních tabulek, případně jejich importu.
- Transformations – komponenta umožňující transformace nad vstupními daty.
- VIMEO – komponenta umožňuje filtrovat data pomocí VIMEO funkcí.
- Reduce – komponenta slouží k redukci množství dat a v případě potřeby dolovací úlohy i rozdělení vstupních dat na trénovací a testovací tabulku.
- Mining module – komponenta slouží jako abstrakce dolovací metody. Místo ní lze použít jakýkoliv dolovací modul.
- Insight – komponentu lze volitelně použít pro zobrazení výstupních dat z jednotlivých částí dolovacího procesu. Neupravuje data.
- Report – komponenta zobrazuje výsledky a informace o procesu dolování. Její obsah je závislý na zvoleném dolovacím modulu.



Obrázek 5.3 GUI - model dolovacího procesu

5.4.3 Dolovací moduly

Modularita systému je postavena na virtuálním souborovém systému implementovaném pomocí XML souboru layer.xml. Zde jsou evidovány existující moduly a tím je umožněno jejich připojování k systému a změna bez nutnosti rekompilace celého jádra. Požadavky na moduly a jejich implementace jsou blíže popsány v kapitole 6 a 7.

5.5 Historie a současný stav

Dolovací nástroj Dataminer je vyvíjen na Fakultě informačních technologií VUT v Brně v rámci diplomových a bakalářských prací od roku 2002. Teprve však od roku 2006 je nástroj převáděn do současné podoby, kdy základy systému položil ve své práci Jádro systému pro dolování z dat v prostředí Oracle pan Doležal [23]. Vznikla tak idea rozdělení systému na jádro a moduly připojené k jádru pomocí definovaného rozhraní. Dolovací proces byl definován pevně stanovenými kroky.

V roce 2007 pokračoval ve vývoji pan Gálet v práci Grafická nadstavba pro systém získávání znalostí. Došlo k výraznému posunu systému z pohledu definice procesu. Ten již není pevně definován, ale lze jej dynamicky tvořit spojováním komponent dolovacího procesu. Dále byla vytvořena většina tříd grafického rozhraní.

Následně v roce 2008 došlo k dokončení definice procesu na úrovni datových struktur pomocí jazyka DMSL v práci pana Krásného s názvem Systém pro dolování z dat v prostředí Oracle.

Počátkem roku 2009, kdy již byl položen ucelený základ jádra systému, se do vývoje zapojilo více lidí především v oblasti dolovacích modulů. Úpravami na jádře systému se zabýval pan Šebek, který se zaměřil především na předzpracování dat v práci Rozšíření funkcionality systému pro dolování z dat na platformě NetBeans [13]. Další práce v tomto roce se již věnovaly jednotlivým dolovacím modulům:

- Mader, P.: Dolovací moduly systému pro dolování z dat v prostředí Oracle - dolovací modul asociačních pravidel založený na ODM založený na algoritmu Apriori. [19]
- Henkel, T.: Dolovací moduly systému pro dolování z dat na platformě NetBeans – klasifikační dolovací modul používající metodu rozhodovacího stromu v ODM. [14]
- Havlíček, D.: Vytvoření nových predikčních modulů v systému pro dolování z dat na platformě NetBeans – predikční modul založený na lineární regresi podporované v ODM pomocí algoritmu SVM. [15]
- Kmoščák, O.: Vytvoření nových klasifikačních modulů v systému pro dolování z dat na platformě NetBeans – klasifikační modul založený na ODM využívající naivní Bayessovskou klasifikaci. [20]

V současné době (rok 2010) pokračují ve vývoji nástroje další lidé především na tvorbě nových dolovacích modulů a celý dolovací nástroj je převáděn na novou verzi databázového serveru Oracle Database 11g Release 2.

5.6 Navrhovaná rozšíření

Na rozšiřitelnost dolovacího nástroje se můžeme podívat z více pohledů. Jako první pohled se nabízí možnost doimplementovat zbývající ještě neimplementované dolovací metody obsažené v ODM nebo

implementace dolovacích algoritmů, které nejsou součástí ODM. Z dalšího pohledu můžeme dolovací nástroj rozšiřovat z hlediska různých druhů netypických dolovacích dat.

5.6.1 Rozšíření o dolovací metody z ODM

Z porovnání dostupných dolovacích funkcí v ODM (kap. 5.2) a jejich současné implementace v Datamineru (kap. 5.5) lze zjistit, že ještě nejsou pokryty všechny nabízené funkce. Pro další rozšíření by tedy bylo vhodné jejich plné využití. Následuje výčet ještě nevyužitých funkcí:

- **Klasifikace** – využití logistické regrese pro další možnost klasifikace.
- **Predikce** – využití vícenásobné lineární regrese založené na Generalized linear models.
- **Detekce odlehlých hodnot** – využití metody klasifikace pomocí one-class SVM.
- **Shlukování** – využití algoritmů k-means a ortogonal partition clustering.
- **Extrakce nových atributů a dolování nejdůležitějších atributů –** pro využití v jiných dolovacích úlohách.

5.6.2 Rozšíření o další dolovací metody

Předem připravené dolovací funkce v ODM nepokrývají všechny známé a zajímavé algoritmy. Z obecných dolovacích a klasifikačních modulů chybí podpora algoritmu FP-Tree pro dolování asociačních pravidel, neboť ODM využívá dle dokumentace Apriori. Dále chybí klasifikace pomocí Bayessovské sítě.

Pro další rozšiřování funkcionality Datamineru tímto směrem bude zajímavé porovnání výkonu nově implementovaných algoritmů na straně klienta s algoritmy obsaženými v ODM na straně serveru.

V oblasti dolování z netypických dat není zatím pokryta žádná oblast, a proto je tedy v tomto směru implementace nových modulů výzvou do dalších let, například v dolování z textu, webu a multimediálních dat.

5.7 Plánovaná rozšíření v této práci

Pro účely pokračování této práce implementací nových algoritmů bylo zvoleno doplnění již existujícího modulu predikce Datamineru o možnost vícenásobné lineární regrese založené na GLM a přidání nového modulu detekce odlehlých hodnot založeného na One-class SVM. Oba tyto doplňky budou využívat existující řešení ODM na straně databázového serveru Oracle Database 11g Release 2.

6 Nový modul detekce odlehlých hodnot

Tato kapitola je věnována vytvoření nového dolovacího modulu pro detekci odlehlých hodnot. V první fázi je zapotřebí nový modul vytvořit a začlenit do stávající aplikace Dataminer. Další fáze vývoje zahrnuje detailní prozkoumání možností algoritmu one-class SVM a rozhraní ODM pro práci s tímto algoritmem. Na základě těchto znalostí je potřeba navrhnout úpravy dokumentu DMSL a v poslední fázi implementovat celý modul.

6.1 Vytvoření a integrace modulu do systému

Protože je aplikace Dataminer vyvíjena na platformě *NetBeans*, je zapotřebí použít vývojovou platformu *NetBeans* s podporou tvorby nových modulů. Při tvorbě nového projektu vybereme možnost *NetBeans Modules - Module*. Zvolíme jméno modulu *AdModule* (Anomaly detection module) a cestu k již existující aplikaci. Modul lze vyvíjet i samostatně, pak je nutné specifikovat stejný typ platformy nového modulu jako využívá celá aplikace. Dále je potřeba specifikovat *Code Base Name* na *cz.vutbr.fit.dataminer*. Nakonec musíme importovat knihovny jádra pro specifikaci projektových závislostí a importování potřebných externích knihoven. Ve složce projektu nového modulu pomocí položky *Properties-Libraries* zvolíme *Add Module Dependency* a zde importujeme *Actions APIs, Dialogs API, dm-api, dm-core-wrapper, dm-editor, UI Utilities API a Utilities API.*

Aby bylo jádro aplikace schopno načíst nový modul a přiřadit ho k již existujícím, je zapotřebí upravit soubor *layer.xml* (obecný popis kapitola 5.4.3), který uchovává informace a parametry o jednotlivých modulech.

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

  <folder name="MiningPieceConfig">
    <folder name="cz-vutbr-fit-dataminer-admodule-AdModule">
      <folder name="accept">
        <file name="cz-vutbr-fit-dataminer-editor-palette-items-Reduce"/>
      </folder>
    </folder>
    <folder name="cz-vutbr-fit-dataminer-editor-palette-items-Report">
      <folder name="accept">
        <file name="cz-vutbr-fit-dataminer-admodule-AdModule"/>
      </folder>
    </folder>
  </folder>
</filesystem>
```

Obrázek 6.1 Obsah layer.xml

Pomocí kódu zobrazeného na obr 6.1 je vytvořena instance třídy AdModule a modul je zaregistrován do MiningPieceRegistry v kategorii Mining Modules. To znamená, že modul bude zobrazen jako komponenta v paletě Mining Modules. Další část záznamu specifikuje možnosti zapojení komponenty modulu ve struktuře dolovacího procesu. Záznam umožňuje připojit komponentu pouze za komponentu *Reduce* a následovat může pouze komponenta *Report*.

Tímto máme modul připravený k jeho implementaci.

6.2 Algoritmus one-class SVM v ODM

Obecný úvod k algoritmu SVM je součástí kapitoly 4.3. Tato kapitola je věnována konkrétní implementaci klasifikačního algoritmu one-class SVM v ODM, která umožňuje plné využití všech výhod a vlastností klasifikace pomocí SVM.

Mezi největší přednosti tohoto klasifikátoru patří schopnost automatického předzpracování dat, efektivní práce s neomezeným počtem atributů, schopnost vyhledat maximální možné nejlepší nastavení parametrů a inkrementální vytváření modelu na malých vzorcích dat, což umožňuje zpracovávat veliké vstupní datové tabulky. Tyto vlastnosti umožňují i běžnému uživateli bez expertních znalostí využívat výhody tohoto algoritmu.

Detekce odlehlých hodnot probíhá ve dvou fázích. V první fázi je algoritmem vytvořen klasifikační model pomocí vstupních dat tvořených pouze relevantními daty a v závislosti na nastavených vstupních parametrech. Ve druhé fázi je tento model aplikován na nová vstupní data a klasifikuje je do dvou tříd. Třída 1 obsahuje relevantní data a třída 0 obsahuje odlehlé hodnoty.

6.2.1 Nastavitelné parametry

Algoritmu one-class SVM implementovaném v ODM lze nastavit následující parametry:

- **Kernel function** – definuje transformační funkci zpracování vstupních dat do vícerozměrného prostoru. Lze nastavit lineární a nelineární (Gaussovský) kernel a tím docílit použití lineárního nebo nelineárního algoritmu SVM. Výběr lze nechat na algoritmu, který vybere vhodný kernel pro zvolená vstupní data.
- **Outlier rate** – hodnota v procentech udávající kolik procent ze vstupních dat bude považováno za odlehlé. Z jiného pohledu tato hodnota znamená orientačně minimální množství hledaných odlehlých hodnot. Jedná se o důležitý parametr při tvorbě modelu schopný výrazně ovlivnit výsledky dolování.
- **Tolerance** – hodnota udává měřítko pro dokončení trénovací fáze tvorby modelu.
- **Active learning** – umožňuje zapnout optimalizační metodu při trénovací fázi tvorby modelu. Optimalizaci je vhodné použít u množství položek nad 100000, protože SVM model vzrůstá s počtem vstupních dat a nad touto hodnotou by už nebylo učení

efektivní zejména pro nelineární kernel, kde zvětšování složitosti a objemu modelu probíhá násobně rychleji.

Následující parametry jsou dostupné pouze pro nelineární (Gaussovský) kernel.

- **Standard deviation** – směrodatná odchylka, která je ve stejném rozsahu jako vzdálenost mezi typickými případy sloužící ke kontrole šíření Gaussovského kernelu. Zadává se jen v případě znalosti hodnoty pro daný druh dat, jinak se nechává spočítat při tvorbě modelu.
- **Cache size** – hodnota omezuje velikost paměti alokované Gaussovským kernelem.

6.2.2 Předzpracování dat

Předzpracování dat je nezbytně nutnou přípravou pro získání kvalitních výsledků algoritmem SVM. Implementace v ODM je schopna sama připravit vstupní data. Pro plnou kontrolu nad předzpracováním dat je však možné předpřipravit data i ručně.

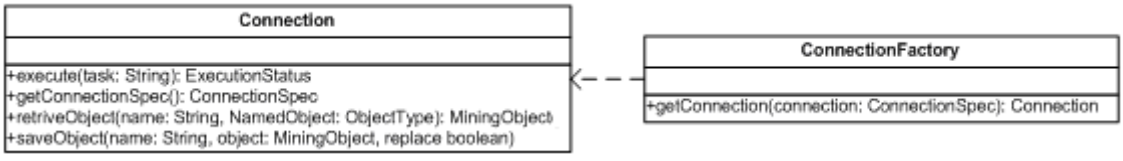
Algoritmus pracuje pouze s hodnotami atributů typu *Numerical*. V první fázi předzpracování dat proto převádí všechny datové typy na čísla. V případě kategorických datových typů nahrazuje postupně hodnoty atributů čísly od nuly po počet kategorií v daném datovém typu.

V následující fázi odstraní prázdné hodnoty. V případě kategorického datového typu je hodnota doplněna hodnotou s největší frekvencí výskytu. V případě numerického typu hodnoty je vložen aritmetický průměr z ostatních hodnot.

Poslední fází je normalizace dat. Při normalizaci dat dojde k upravení hodnot numerických atributů do intervalu <0,1>. Tato fáze předzpracování dat je dostupná až od verze Oracle 11g. V předchozích verzích bylo nutné normalizaci provést vždy ručně.

6.2.3 Rozhraní ODM pro práci s one-class SVM

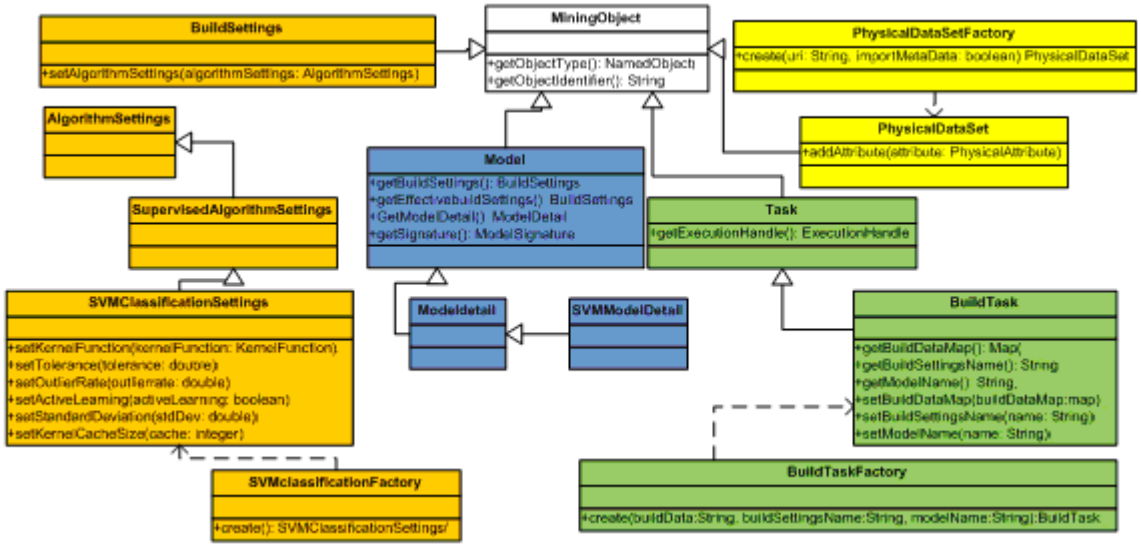
Rozhraní ODM umožňuje vytvoření klasifikačního modelu pro detekci odlehlých hodnot a následně i aplikaci takto vytvořeného modelu. Vytváření instancí potřebných tříd pro práci s modelem je prováděno přes metodu *create()* příslušných tříd Factory. Nejprve je nutné vytvořit připojení k databázi, na které bude probíhat dolovací úloha pomocí třídy Connection. Tato třída zajišťuje veškerou komunikaci (spouštění úloh, ukládání objektů, tvorba a práce s modelem) s databází.



Obrázek 6.2 Třídy pro vytvoření spojení s databází

6.2.3.1 Fáze vytvoření modelu

Obrázek 6.3 znázorňuje třídy pro vytvoření klasifikačního modelu pro detekci odlehlých hodnot. Prvním krokem je vytvoření objektu PhysicalDataSet určujícího strukturu vstupních dat vytvářejících model (znázorněno žlutou barvou). Je nutné specifikovat nejen název vstupní tabulky, ale i primární klíč vstupní tabulky. Pro nastavení parametrů dolovacího algoritmu slouží objekt třídy BuildSettings přijímající objekt SVMClassificationSettings s nastavenými parametry pro SVM klasifikátor (znázorněno oranžovou barvou). Objekt Task slouží pro specifikaci činností, které bude v určité fázi dolovací engine provádět (vytvoření modelu, aplikace modelu). Vytvoření modelu proběhne pomocí objektu BuildTask (znázorněno zelenou barvou) a detaily vzniklého modelu jsou přístupny přes objekt Model, popřípadě SVMModelDetail.

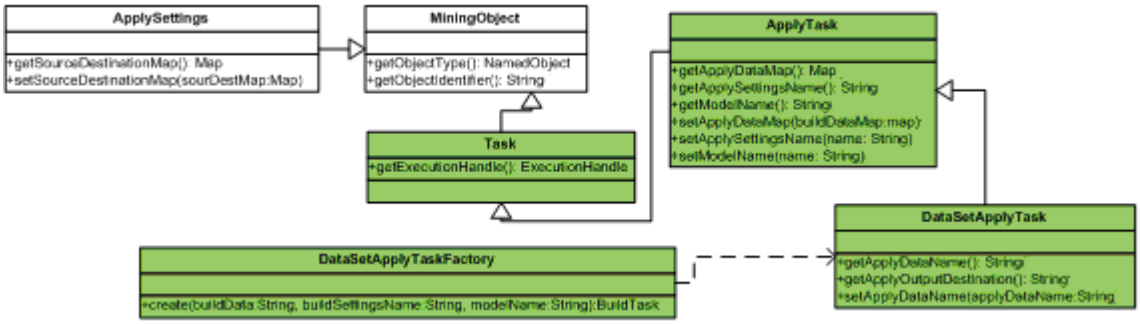


Obrázek 6.3 Zjednodušený přehled tříd pro vytvoření klasifikačního modelu

Detailní informace k třídám ODM lze dohledat na [22].

6.2.3.2 Fáze aplikace modelu na nová data

Při fázi aplikace modelu dochází k použití již existujícího modelu vytvořeného v minulé kapitole na novou tabulku dat. Je zapotřebí specifikovat nejen vstupní tabulku objektem PhysicalDataSet stejným způsobem jako v předchozí fázi, ale i výstupní tabulku pro zápis výsledků dolování v objektu DataSetApplyTask. Pro samotnou aplikaci slouží objekt ApplyTask. Výsledek dolování je pak dostupný dotazem na tabulku specifikovanou v DataSetApplyTask.



Obrázek 6.4 Zjednodušený přehled tříd pro aplikaci klasifikačního modelu

6.3 Úpravy v DMSL

Pro dolovací modul detekce odlehlých hodnot jsou podstatné dva elementy v dokumentu DMSL. Element DataMiningTask specifikuje typ a parametry dolovací úlohy a element Knowledge uchovává informace o samotném dolování a jeho výsledcích. Oba tyto elementy mají volnou syntaxi a je zapotřebí je přizpůsobit na míru novému modulu detekce odlehlých hodnot, který využívá algoritmu one-class SVM obsaženého v ODM.

Tento algoritmus v úpravě pro regresní analýzu využívá i práce pana Havlíčka [15], a proto se jeví jako velice vhodné využít pro základ struktury obou elementů jeho návrhu, který se dá jednoduše upravit i pro klasifikační verzi one-class SVM.

6.3.1 Element DataMiningTask

DataMiningTask je kořenový element uchovávající typ algoritmu detekce odlehlých hodnot a jeho parametry pomocí vnitřního elementu AnomalyDetection. Obsahuje povinný atribut *name* sloužící k propojení všech elementů spadajících pod jednu dolovací úlohu a další povinný parametr *language* udávající jazyk, ve kterém je element DataMiningTask popsán. K popisu elementu je zvolen jazyk XML.

```
<!ELEMENT DataMiningTask (AnomalyDetection)>
<!ATTLIST DataMiningTask name CDATA #REQUIRED
                         language CDATA #REQUIRED
                         languageVersion CDATA #IMPLIED
                         type CDATA #IMPLIED>
```

Element AnomalyDetection je navržen tak, aby jednoznačně určoval povinným atributem *algorithm* použitý algoritmus pro dolování odlehlých hodnot.

```
<!ELEMENT AnomalyDetection (Usematrix, InputDataType, Parametres)>
<!ATTLIST AnomalyDetection algorithm CDATA #REQUIRED>
```

Zdroj dat pro algoritmus obsahuje element Usematrix s atributem *matrixRef* obsahujícím název zdrojové tabulky. Element InputDataType specifikuje povinným atributem *tableKey* primární klíč

zdrojové tabulky. Pro dolovací algoritmus One-Class SVM není nutné uvádět atribut *target*, který je součástí původního návrhu pana Havlíčka [15], a proto je vypuštěn.

```
<!ELEMENT Usematrix>
<!ATTLIST Usematrix matrixRef CDATA #REQUIRED>
<!ELEMENT InputDataType>
<!ATTLIST InputDataType tableKey CDATA #REQUIRED>
```

Element Parametres je celý přepracován tak, aby uchovával v elementech všechny parametry důležité pro detekci odlehlých hodnot. Názvy elementů odpovídají názvům parametrů one-class SVM algoritmu a jsou blíže popsány v kapitole 6.2.1. Elementy StandardDeviation a CacheSize nejsou povinné.

```
<!ELEMENT Parametres (KernelFunction, Tolerance, ActiveLearning, OutlierRate,
                                        StandardDeviation?, CacheSize?)>
<!ELEMENT KernelFunction (Gaussian|Linear|System determined)>
<!ELEMENT OutlierRate (%REAL-NUMBER;)>
<!ELEMENT ActiveLearning (true|false)>
<!ELEMENT Tolerance (%REAL-NUMBER;)>
<!ELEMENT StandardDeviation (%REAL-NUMBER;)>
<!ELEMENT CacheSize (%INT-NUMBER;)>
```

6.3.2 Element Knowledge

Knowledge je kořenovým elementem pro uchování informací o jednotlivých fázích dolování odlehlých hodnot. Návrh elementu je celý převzat z práce pana Havlíčka [15], ze kterého jsou jednotlivé části upraveny podle odlišností ve výstupech regresního SVM a one-class SVM algoritmu.

Element Knowledge obsahuje povinný atribut *name* sloužící k propojení všech elementů spadajících pod jednu dolovací úlohu a další povinný parametr *language* udávající jazyk, ve kterém je element DataMiningTask popsán. K popisu elementu je zvolen jazyk XML.

```
<!ELEMENT Knowledge (BuildTask, ApplyTask)>
<!ATTLIST Knowledge name CDATA #REQUIRED
                    language CDATA #REQUIRED
                    LanguageVersion CDATA #IMPLIED
                    type CDATA #IMPLIED >
```

Proces dolování odlehlých hodnot obsahuje dvě fáze - vytvoření klasifikačního modelu a aplikaci na nová data. Tyto fáze jsou zaznamenány v elementech BuildTask a ApplyTask. Fáze testování není součástí procesu dolování odlehlých hodnot, proto je vypuštěna.

Element BuildTask zachycuje informace o vytvořeném klasifikačním modelu. Uchovává v sobě v atributu *name* název komponenty, která model vytvořila, dále jméno vytvořeného klasifikačního modelu v atributu *model*, název vstupní tabulky pro vytvoření modelu v atributu *input* a informace o vytvoření, ukončení úlohy a výsledném stavu úlohy v atributech *start*, *end*, *status*.

```
<!ELEMENT BuildTask (Model)>
<!ATTLIST BuildTask name CDATA #REQUIRED
                                    model CDATA #REQUIRED
                                    input CDATA #REQUIRED
                                    start %DATE-TIME; #REQUIRED
                                    end %DATE-TIME; #REQUIRED
                                    status CDATA; #REQUIRED >
```

O vytvořeném klasifikačním modelu jsou uloženy informace v elementu Model, který obsahuje identifikační atribut *name* s názvem modelu a typ modelu uložený v atributu *type*. Informace o sloupcích tabulky, ze kterých je model vytvářen, jsou uloženy v elementu ModelSignatureAttribute, který uchovává v atributech *name*, *dataType* a *miningType* informace o názvu sloupce, jeho datovém typu a typu dolovacích dat, které obsahuje.

```
<!ELEMENT Model (ModelSignatureAttribute)>
<!ATTLIST Model name CDATA #REQUIRED
                        type (Linear|Gaussian) #REQUIRED >
<!ELEMENT ModelSignatureAttribute >
<!ATTLIST ModelSignatureAttribute name CDATA #REQUIRED
                                    dataType CDATA #REQUIRED
                                    miningType CDATA #REQUIRED >
```

Poslední element ApplyTask obsahuje informace o aplikační fázi dolování odlehlých hodnot. Element je nepovinný a je použit pouze, když k této fázi dojde. Obsahuje povinné atributy *name*, *model*, *start*, *end*, *status* jako model BuildTask a dále atribut *input* s názvem vstupní tabulky pro aplikaci klasifikačního modelu a atribut *output* obsahující název výstupní tabulky s výsledky aplikační fáze.

```
<!ELEMENT ApplyTask EMPTY>
<!ATTLIST ApplyTask name CDATA #REQUIRED
                        model CDATA #REQUIRED
                        input CDATA #REQUIRED
                        output CDATA #REQUIRED
                        start %DATE-TIME; #REQUIRED
                        end %DATE-TIME; #REQUIRED
                        status CDATA; #REQUIRED >
```

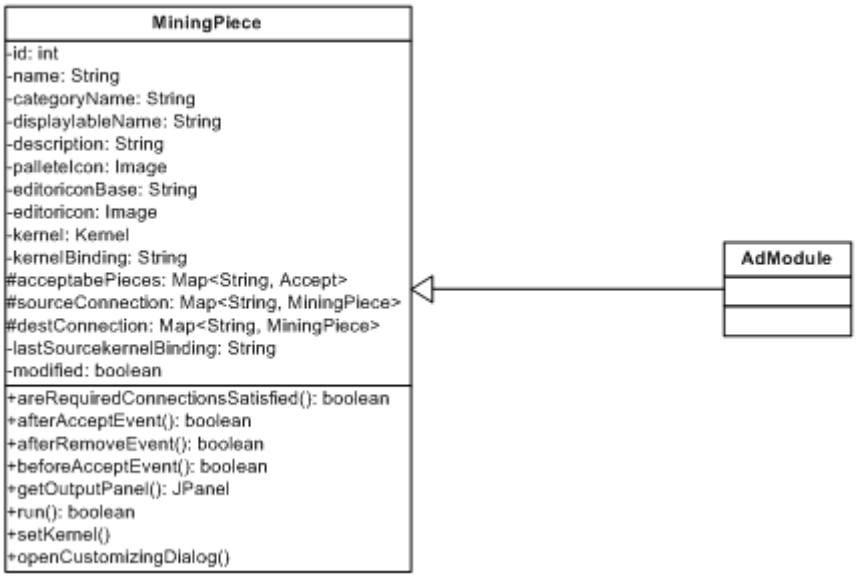
6.4 Implementace modulu

Implementace modulu detekce odlehlých hodnot v sobě zahrnuje implementaci abstraktní třídy *MiningPiece* a všech jejích metod, implementaci práce s částmi DataMiningTask a Knowledge dokumentu DMSL, implementaci vytvoření dolovacího modelu, jeho aplikaci na nová data a zpracování výsledků dolování.

Při implementaci je zapotřebí v různých fázích dolování dat udržovat konzistentní záznam o procesu dolování ve strukturách DMSL a v jakékoliv fázi musí být možné stav úlohy uložit. DMSL dokument je jediné místo pro uchování těchto informací, a proto bude vždy u každého popisu implementace funkcionality uveden vliv na data v něm uložená.

6.4.1 Implementace třídy MiningPiece

Každá komponenta, která je součástí dolovacího procesu a má být přístupná v aplikaci, musí implementovat abstraktní třídu MiningPiece nacházející se v API vrstvě. Třída nazvaná AdModule, která tuto třídu implementuje, se stává hlavní třídou modulu pro detekci odlehlých hodnot.



Obrázek 6.5 Implementace třídy MiningPiece

V konstruktoru třídy je nutné nastavit jméno, popis a cesty k ikonám komponenty. Ikonu v paletě komponent nastaví setPaletteIcon a ikonu používanou v editoru nastaví setEditorIcon. Jméno a popis modulu je nastaveno pomocí lokalizačního bundle. Tento postup umožňuje případnou hromadnou lokalizaci všech textů použitých v grafickém rozhraní modulu soustředěných do souboru *Bundle.properties*.

```
setDisplayableName(NbBundle.getMessage(AdModule.class, "NAME_AdModule"));
setDescription(NbBundle.getMessage(AdModule.class, "HINT_AdModule"));
setPaletteIcon("cz/vutbr/fit/dataminer/admodule/resources/AdModule16.png");
setEditorIcon("cz/vutbr/fit/dataminer/admodule/resources/AdModule48.png");
```

Obrázek 6.6 Nastavení v konstruktoru třídy AdModule implementující MiningPiece

Ve třídě AdModule je dále zapotřebí implementovat všechny metody obsažené v abstraktní třídě MiningPiece (obr 6.5), aby bylo možno s komponentou nového modulu v systému pracovat. Následuje jejich výčet a popis, kdy jsou metody volány:

- **beforeAcceptEvent()** – metoda je volána před vložením dolovací komponenty na plochu editoru dolovacího procesu;

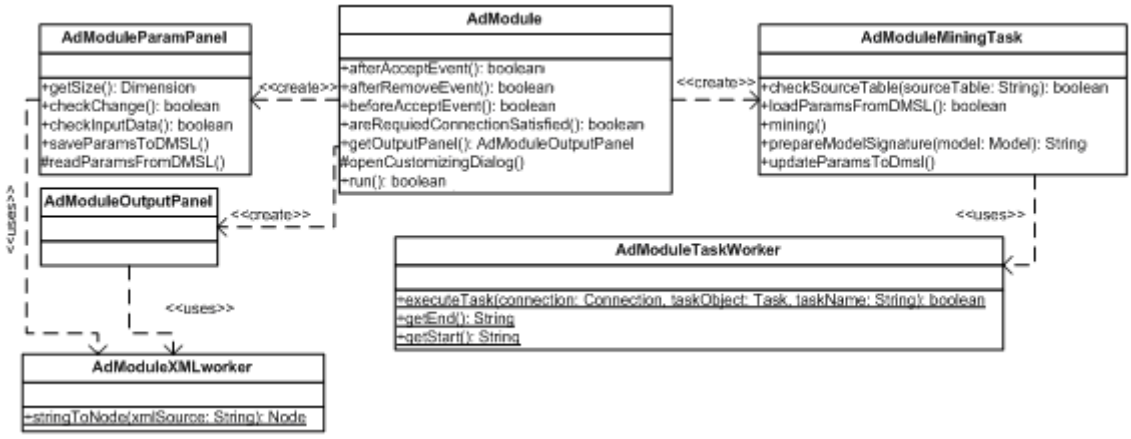
- **afterAcceptEvent()** – metoda je volána po vložení dolovací komponenty na plochu editoru dolovacího procesu;
- **afterRemoveEvent()** – metoda je volána při odstranění dolovací komponenty z plochy editoru dolovacího procesu;
- **openCustomizingDialog()** – metoda je zavolána při požadavku *open* nad dolovací komponentou; slouží pro zobrazení dialogu zadávání parametrů komponenty;
- **run()** – metoda je volána při požadavku *run* nad vloženou dolovací komponentou; provede dolovací úlohu;
- **getOutputPanel()** – metoda slouží k vytvoření panelu pro zobrazení výsledků proběhlé dolovací úlohy; je volána při požadavku *open* komponenty *Report*.

Následující kapitoly se zabývají konkrétní implementací jednotlivých metod v modulu detekce odlehlých hodnot.

6.4.2 Struktura modulu

Hlavní třídou modulu je třída AdModule implementující abstraktní třídu MiningPiece. Tato třída zajišťuje komunikaci s jádrem aplikace a tvoří zastřešující třídu ke všem třídám a metodám zajišťující funkcionalitu celého modulu.

Na obrázku 6.7 je znázorněna základní struktura celého modulu. Třída AdModuleParamPanel implementuje zadávání parametrů dolovací úlohy. Třída AdModuleOutputPanel implementuje zobrazení výsledků dolování odlehlých hodnot a informace o klasifikačním modelu. Výše zmíněné třídy využívají třídu AdModuleXMLworker pro vytvoření uzlu ve struktuře dokumentu DMSL.



Obrázek 6.7 Struktura modulu pro dolování odlehlých hodnot

Třída AdModuleMiningTask implementuje vytvoření klasifikačního modelu pro detekci odlehlých hodnot a využívá třídu AdModuleTaskWorker obstarávající samostatnou úlohu tvorby modelu.

6.4.3 Vložení komponenty modulu

Vložení komponenty do editoru dolovacího procesu je implementováno pomocí metody *afterAcceptEvent()*. Ta uloží základní informace a nastavení o modulu do DMSL (obr. 6.8) pomocí elementů DataMiningTask a Knowledge s atributem *name* odpovídajícím názvu komponenty.

Parametry algoritmu v elementu DataMiningTask jsou nastaveny na základní a je nutné je před spuštěním dolování doplnit pomocí panelu parametrů popsaném v následující kapitole. Atributy všech elementů v Knowledge jsou nastaveny na počáteční hodnotu *null*.

V případě odstranění komponenty jsou elementy DataMiningTask a Knowledge s atributem *name* odpovídajícím jménu odstraněné komponenty odstraněny z dokumentu DMSL pomocí metody *afterRemoveEvent()*.

Pokud dojde ke vložení více komponent stejného modulu na plochu editoru dolovacího procesu, je každé vygenerováno unikátní číslo v metodě *beforeAcceptEvent()*.

```
<DataMiningTask  language="XML" name="DM633_AdModule3" type="AnomalyDetection">
 <AnomalyDetection  algorithm="One-Class SVM">
  <Usematrix  matrixRef="null"></Usematrix>
  <InputDataType  tableKey="null"></InputDataType>
  <Parametres >
    <KernelFunction >System determined </KernelFunction>
    <OutlierRate >0.1 </OutlierRate>
    <Tolerance >0.001 </Tolerance>
    <ActiveLearning >False </ActiveLearning>
 </Parametres>
 </AnomalyDetection>
</DataMiningTask>

<Knowledge  language="XML" name="DM633_AdModule3" type="AnomalyDetection">
 <BuildTask  end="null" input="null" model="null" name="null" start="null" status="null">
 </BuildTask>
 <ApplyTask  end="null" input="null" model="null" name="null" output="null" start="null" status="null">
 </ApplyTask>
</Knowledge>
```

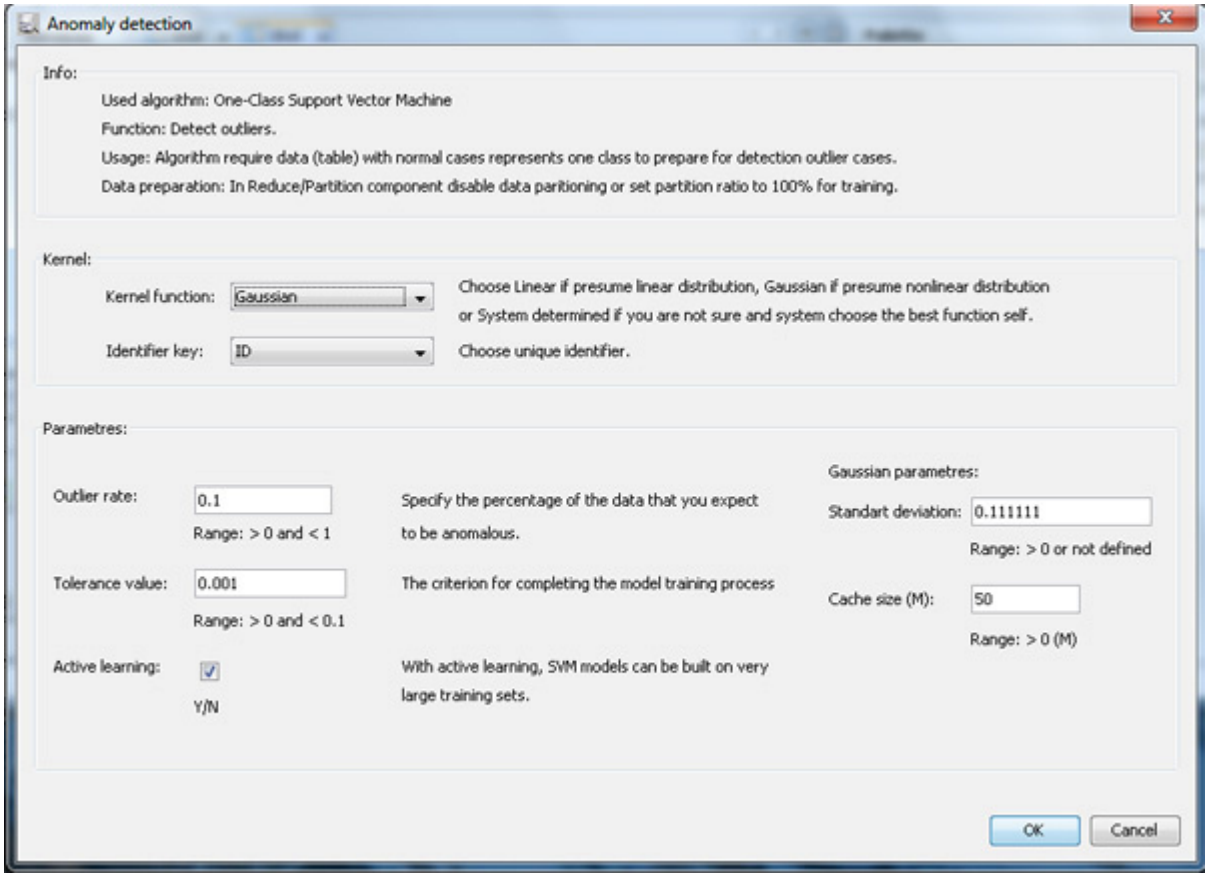
Obrázek 6.8 Základní stav DMSL po vložení komponenty do editoru dolovacího procesu

6.4.4 Panel parametrů

Po vyvolání metody *openCustomizingDialog()* je otevřen panel pro zadání parametrů dolovací úlohy, který je implementován třídou AdModuleParamPanel. Zobrazení je povoleno pouze v případě, pokud je komponenta připojena ke zdroji dat. Při zobrazení jsou načteny hodnoty parametrů z obsahu elementu DataMiningTask. Při zavření panelu je provedena kontrola validity zadaných dat a v případě úspěchu jsou parametry uloženy do DMSL. V případě neúspěšné kontroly je uživatel upozorněn na chybnou hodnotu a vyzván k opravě.

Panel parametrů se skládá ze tří částí. První část je čistě informativní. Druhá část umožňuje nastavit kernel SVM algoritmu a sloupec s primárním klíčem vstupní tabulky. Ve třetí části lze nastavit všechny parametry algoritmu. Parametry Standard deviation a Cache size jsou nastavitelné

pouze při výběru Gaussovského kernelu. Výčet nastavitelných parametrů one-class SVM algoritmu je součástí kapitoly 6.2.1.



Obrázek 6.9 Panel pro zadávání parametrů dolovací úlohy detekce odlehlých hodnot

6.4.5 Spuštění dolování

Metoda *run()* spustí vytvoření klasifikačního modelu pro detekci odlehlých hodnot. Tuto funkcionalitu implementuje třída AdModuleMiningTask. Informace o vstupní tabulce jsou brány z komponenty Reduce/Partition. Nastavení parametrů one-class SVM je načteno z DMSL elementu DataMiningTask. Třída pracuje s rozhraním ODM popsaným v kapitole 6.2.3.1. Samotná úloha vytvoření klasifikačního modelu je implementována pomocí třídy AdModuleTaskWorker.

Následně po vytvoření modelu jsou upraveny a uloženy změny vstupních parametrů do DMSL. Informace o vytvořeném klasifikačním modelu jsou uloženy do DMSL elementu buildTask.

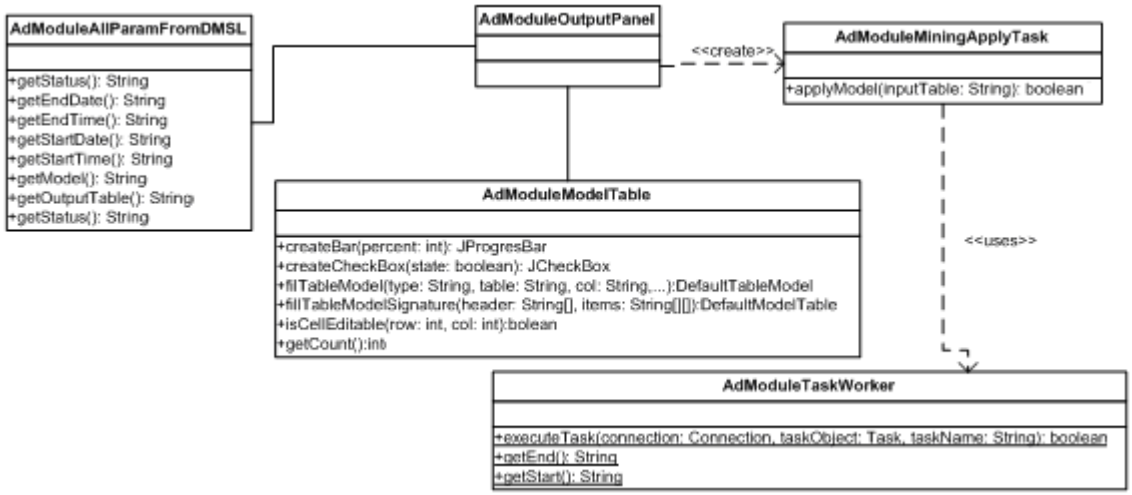
6.4.6 Prezentace výsledků

Vytvořený klasifikační model je prezentován pomocí komponenty Report, která je implementována třídou AdModuleOutputPanel a je volána metodou *getOutputPanel()*.

Jako základ pro grafický návrh panelu zobrazení výsledků detekce odlehlých hodnot byl použit návrh pana Havlíčka z jeho práce implementující regresní modul využívající algoritmus SVM [15].

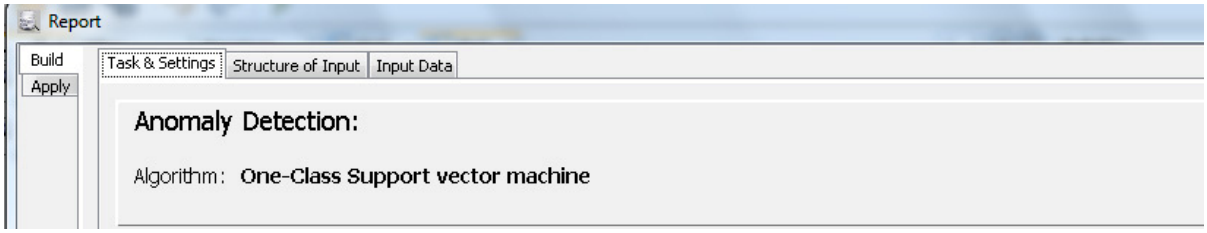
Tato možnost byla využita z důvodu sjednocení uživatelského rozhraní obou modulů využívající v základu stejného algoritmu. Návrh pana Havlíčka byl rozvinut a upraven pro potřeby klasifikačního modelu. Byla odstraněna fáze testování modelu a součásti související s regresním modelem. Aplikační fáze byla přepracována tak, aby splňovala potřeby detekce odlehlých hodnot.

Schéma implementace je zobrazeno na obrázku 6.10. Třída AdModuleModelTable slouží k přípravě všech tabulek. Třída AdModuleAllParamFromDMSL načítá dostupné informace o klasifikačním modelu. Součástí prezentace výsledků je i aplikační fáze detekce odlehlých hodnot implementovaná třídou AdModuleMiningApplyTask.



Obrázek 6.10 Implementace komponenty Report

Fáze dolování odlehlých hodnot jsou ve výstupním panelu reprezentovány pomocí dvou záložek Build a Apply. Záložka Build obsahuje všechny informace o vytvořeném klasifikačním modelu. Informace jsou čerpány z DMSL elementu buildTask. Záložka Apply slouží k aplikaci klasifikačního modelu na nová data a k zobrazení výsledků aplikační fáze.



Obrázek 6.11 Grafická podoba záložek Build a Apply

Práce s komponentou detekce odlehlých hodnot a zobrazením výsledků dosažených dolováním je vysvětlena na konkrétním příkladě v kapitole 8.

7 Rozšíření modulu predikce

Modul predikce navrhl a implementoval pro algoritmus SVM ve své práci Vytvoření nových predikčních modulů v systému pro dolování z dat na platformě NetBeans pan Havlíček [15]. Tato práce navazuje na jeho výsledky a rozšiřuje je o algoritmus vícenásobné lineární regrese pomocí GLM.

Cílem této kapitoly je popis návrhu rozšíření modulu o algoritmus GLM, jeho implementace a začlenění do existujícího modulu. Nezbytnou součástí návrhu je úprava DMSL dokumentu pro účely uložení parametrů nového algoritmu a získaných znalostí. Další součástí rozšíření modulu je začlenění podpory dolování dat pomocí ODM pro nový algoritmus a jeho samotná implementace. Současně je nutné provést integraci nového algoritmu do stávající implementace tak, aby bylo možné používat oba algoritmy bez újmy na jejich výsledné funkčnosti a možnostech nastavení parametrů.

7.1 Algoritmus GLM v ODM

Obecnému úvodu do zobecněných lineárních modelů - GLM je věnována kapitola 3.4. V této kapitole je popsána implementace algoritmu založeného na GLM v Oracle Data Mining za účelem predikce pomocí vícenásobné lineární regrese.

Algoritmus je součástí ODM až od verze Oracle databáze 11g. Vychází z důkladné analýzy vstupních dat jednak na úrovni celé tabulky, jednotlivých atributů (sloupců) i řádků, při které zjišťuje, nakolik jsou splněny základní podmínky klasického lineárního modelu. V případě nesplnění základních podmínek se snaží využít známých zobecnění podmínek a vytvořit tak model nejlépe odpovídající vstupním datům. Nejčastěji dochází k upravení podmínek na základě detekce jiného rozložení dat vstupních atributů než normálního.

Algoritmus umí pracovat s velkým množstvím vstupních dat a atributů, přičemž jediným omezením při tvorbě modelu jsou dostupné systémové prostředky.

7.1.1 Nastavitelné parametry

Algoritmu je možné nastavit tyto vstupní parametry:

- **Confidence level** – algoritmus GLM je schopný určit hranice intervalu, ve kterém se nachází predikovaná hodnota s jistou pravděpodobností určenou touto hodnotou.
- **Ridge regresion** – technika, která redukuje multikolinearitu vstupních atributů. Multikolinearita znázorňuje lineární závislosti vstupních atributů. Nejlepší výsledky podává algoritmus s nejmenší multikolinearitou. Je možné ji povolit nebo zakázat.

V případě použití Ridge regression jsou zpřístupněna nastavení dalších dvou jejích parametrů:

- **Ridge value** – koeficient Ridge regresion pro redukci multikolinearity. Pokud není nastaven, je automaticky generován nejlepší vhodný.
- **VIF** – nastavení příznaku Variance Inflation Factor počítajícího statistiky ze vstupních dat při vzniku regresního modelu. Statistiky jsou počítány pouze při dostatku systémových zdrojů.

7.1.2 Předzpracování dat

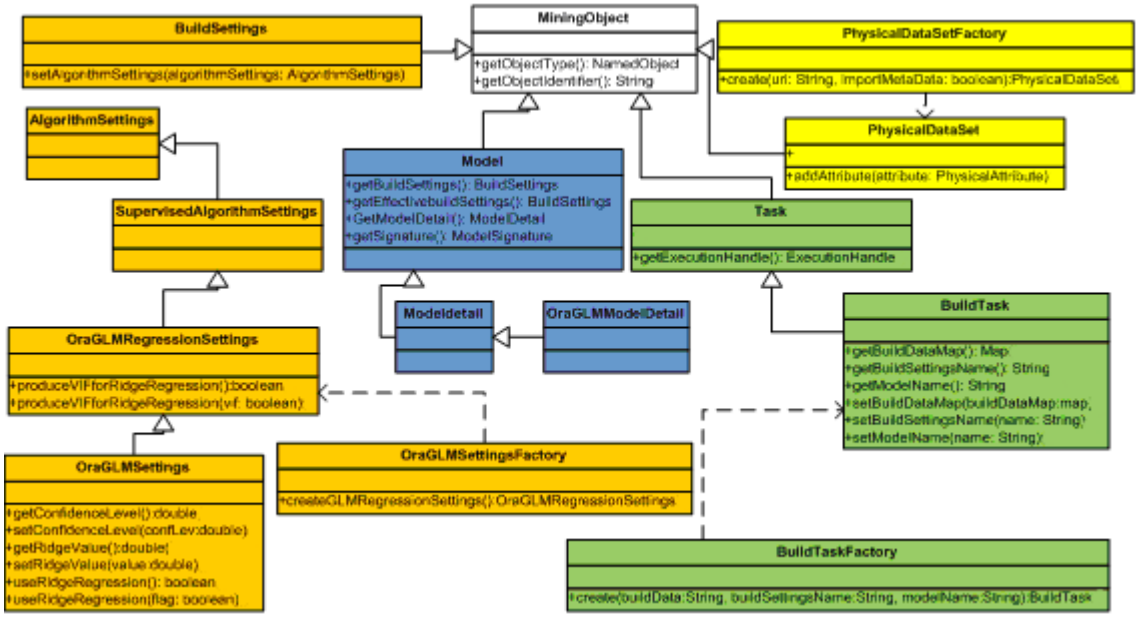
Oracle doporučuje využít automatizované předzpracování vstupních dat pro algoritmus GLM, a proto je v následující implementaci využito. Automatické předzpracování je implicitně zapnuté a nemusí se povolovat. Jeho součástí je doplnění chybějících hodnot, standardizace dat pomocí korelačních transformací a normalizace.

7.1.3 Rozhraní ODM pro práci s GLM

Rozhraní ODM umožňuje pracovat s algoritmem GLM ve třech fázích. První fáze vytváří regresní model podle zadaných parametrů algoritmu a vstupních dat pro trénování. Druhá fáze vytvořený model otestuje pomocí dat určených k testování a vrátí výsledky testů. Třetí fáze umožňuje získaný regresní model aplikovat na nová vstupní data. Práce s databází je shodná jako v případě modulu pro detekci odlehlých hodnot (kapitola 6.2.3).

7.1.3.1 Fáze vytvoření modelu

Oranžová část diagramu tříd na obrázku 7.1 tvoří část pro nastavení parametrů algoritmu GLM pomocí tříd OraGLMSettings a OraGLMRegressionSettings, kde je také specifikován predikovaný atribut.

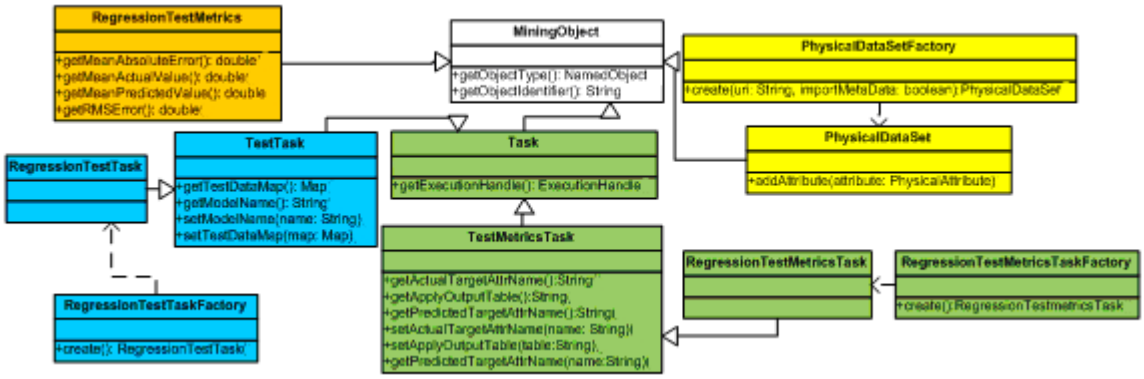


Obrázek 7.1 Zjednodušený přehled tříd pro tvorbu regresního modelu

Vstupní data jsou definována pomocí žlutě označených tříd PhysicalDataSet a PhysicalDataSetFactory. Samotné vytvoření modelu je pokryto v třídě BuildTask, nacházející se v zeleně označené části, a informace o vytvořeném objektu jsou přístupné přes třídu v modře označené části diagramu Model, popřípadě OraGLMModelDetail.

7.1.3.2 Fáze testovací

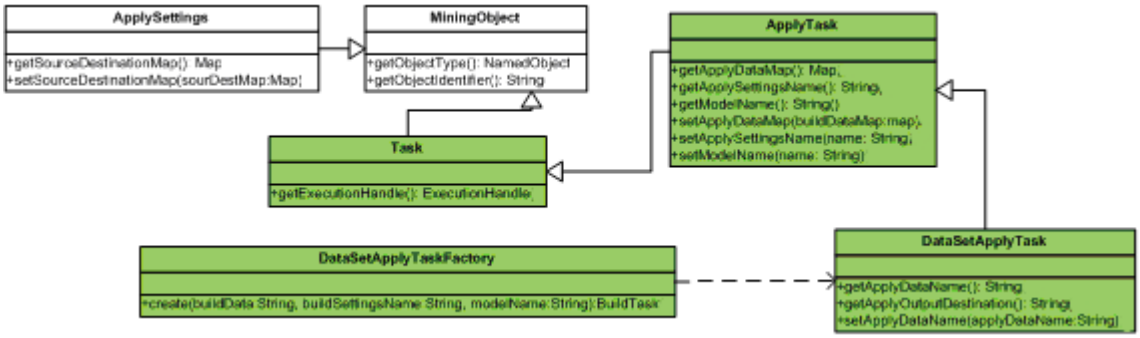
Při testovací fázi jsou pomocí třídy PhysicalDataSet definována testovací data k otestování regresního modelu vzniklého v předchozí fázi. Testování pomocí takto definovaných dat probíhá pomocí třídy TestTask (modrá část diagramu na obrázku 7.2). O výpočet testovacích metrik se stará třída TestMetricsTask (zelená část diagramu) a hodnoty takto získané jsou dostupné přes třídu RegressionTestMetrics (oranžová část diagramu). Testování regresního modelu slouží ke zjištění kvality jeho predikce.



Obrázek 7.2 Zjednodušený přehled tříd pro testování regresního modelu

7.1.3.3 Fáze aplikační

Cílem aplikační fáze je aplikace vytvořeného regresního modelu na nová data, tedy získat z nich hodnoty neznámého predikovaného atributu. O aplikaci modelu se stará třída ApplyTask, kde je definována výstupní tabulka s výsledky predikce (zelená část v diagramu na obrázku 7.3). Vstupní tabulka s novými daty je definována pomocí třídy PhysicalDataSet jako v předchozích fázích.



Obrázek 7.3 Zjednodušený přehled tříd pro aplikaci regresního modelu

7.2 Rozšíření DMSL

Základní struktura DMSL je definována v práci pana Havlíčka [15]. Návrh je potřeba pouze rozšířit o možnosti uložení parametrů algoritmu GLM do elementu DataMiningTask a rozšířit možnost uložení informací o regresním modelu v elementu BuildTask, který je součástí informací uložených do elementu Knowledge.

7.2.1 Element DataMiningTask

Rozšíření elementu DataMiningTask dokumentu DMSL spočívá v úpravě elementu Regression, který se stává jediným místem, kde je uložen vybraný algoritmus k dolování. Typ algoritmu je určen zkratkou SVM nebo GLM.

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

<!ELEMENT Regression (InputDataType, Parametres)>
<!ATTLIST Regression algorithm (SVM, GLM) #REQUIRED>
```

Z původního návrhu je důležité upozornit na atribut *target* elementu InputDataType, který specifikuje název predikovaného sloupce ze vstupní tabulky dat.

```
<!ELEMENT InputDataType>
<!ATTLIST InputDataType tableKey CDATA #REQUIRED
                         target CDATA #REQUIRED>
```

Do elementu Parametres jsou přidány vnitřní elementy pro uložení parametrů algoritmu GLM. Element RidgeRegression je nepovinný. V případě, že není použit, je možnost Ridge regression v algoritmu zakázána. Jeho atribut *VIF* slouží k uložení příznaku zapnutí rozšířených statistik algoritmu GLM. Vnitřní element RidgeValue obsahuje hodnotu stejnojmenného parametru, pokud je zadán.

```
<!ELEMENT Parametres (KernelFunction?, Tolerance?, ActiveLearning?, Epsilon?,
        ComplexityFactor?, StandardDeviation?, CacheSize? | RidgeRegression?,
        ConfidenceLevel?)>
<!ELEMENT KernelFunction (Gaussian|Linear|System determined)>
<!ELEMENT Tolerance (%REAL-NUMBER;)>
<!ELEMENT ActiveLearning (true|false)>
<!ELEMENT Epsilon (%REAL-NUMBER;)>
<!ELEMENT ComplexityFactor (%REAL-NUMBER;)>
<!ELEMENT StandardDeviation (%REAL-NUMBER;)>
<!ELEMENT CacheSize (%INT-NUMBER;)>
<!ELEMENT ConfidenceLevel (%REAL-NUMBER;)>
<!ELEMENT RidgeRegression (RidgeValue?)>
<!ATTLIST RidgeRegression VIF (true|false) #REQUIRED>
<!ELEMENT RidgeValue (%REAL-NUMBER;)>
```

7.2.2 Element Knowledge

Element Knowledge slouží k uložení informací o regresním modelu a získaných znalostí jeho aplikací na nová data. Skládá se ze tří hlavních elementů popisujících tři fáze predikce a je převzat z práce pana Havlíčka [15].

První element BuildTask uchovává informace o vzniku regresního modelu (vstupní tabulka, čas vytvoření, čas dokončení modelu, výsledný status tvorby modelu) a informace o něm ve vnitřních elementech Model a ModelSignature, kde jsou uloženy názvy sloupců vstupní tabulky s typy dat v nich uložených. Element ModelSignature obsahuje vnitřní element AttributeProperties, který v atributu *coefficient* uchovává vypočítanou hodnotu regresního koeficientu. Element BuildTask je rozšířen o element Params pro možnost uložení statistických ukazatelů získaných při tvorbě GLM modelu. Jejich počet a složení nemusí být předem známo. Ukazatele jsou do něj ukládány v následujícím formátu. Název elementu odpovídá názvu statistického ukazatele a jeho hodnota odpovídá hodnotě ukazatele.

```
<!ELEMENT BuildTask Model>
<!ATTLIST BuildTask name CDATA #REQUIRED
                    model CDATA #REQUIRED
                    input CDATA #REQUIRED
                    start %DATE-TIME; #REQUIRED
                    end %DATE-TIME; #REQUIRED
                    status CDATA; #REQUIRED >
<!ELEMENT Model (ModelSignatureAttribute)>
<!ATTLIST Model name CDATA #REQUIRED
                    bias %REAL-NUMBER #IMPLIED
                    isLinear (true|false) #REQUIRED >
<!ELEMENT ModelSignatureAttribute (AttributeProperties*) >
<!ATTLIST ModelSignatureAttribute name CDATA #REQUIRED
                    dataType CDATA #REQUIRED
                    miningType CDATA #REQUIRED >
<!ELEMENT AttributeProperties EMPTY>
<!ATTLIST AttributeProperties value CDATA #REQUIRED
                    coefficient %REAL-NUMBER; #REQUIRED >
<!ELEMENT Params ANY>
```

Element TestTask slouží k uchování informací o testovací fázi a uložení metrik v atributech elementu Metrics znázorňujících kvalitu vytvořeného predikčního modelu. Popis metrik je uveden v kapitole 8.1.

```
<!ELEMENT Metrics EMPTY>
<!ATTLIST Metrics MAE %REAL-NUMBER; #REQUIRED
                    MAV %REAL-NUMBER; #REQUIRED
                    MPV %REAL-NUMBER; #REQUIRED
                    RMSE %REAL-NUMBER; #REQUIRED >
```

Element ApplyTask uchovává v atributech informace o vstupní tabulce pro aplikaci modelu, informace o výstupní tabulce s výsledky predikce a stavu aplikační fáze. Podrobná specifikace je dostupná v práci pana Havlíčka [15].

7.3 Implementace rozšíření modulu

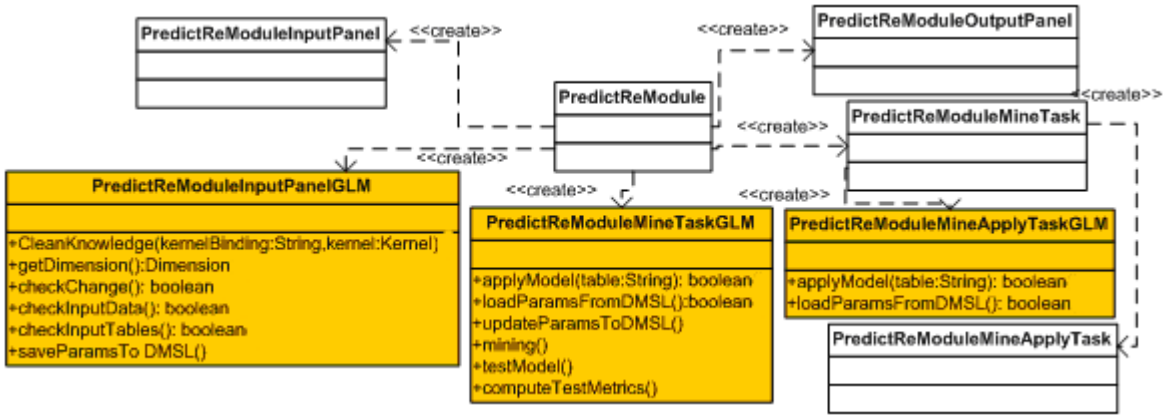
Implementace rozšiřuje již existující regresní modul a v případě možnosti bylo snahou využít již existujícího řešení s úpravou pro nový algoritmus hlavně v části prezentace výsledků, kdy výsledky obou predikčních algoritmů jsou z velké části podobné.

Nejprve ještě před samotnou implementací rozšíření modulu bylo potřeba oddělit již existující programové konstrukce od závislosti na SVM algoritmu. V celém řešení modulu bylo vždy předpokládáno použití pouze jednoho predikčního algoritmu. Tento problém byl převážně vyřešen metodou getRegressionAlgorithm() přidanou do hlavní třídy predikčního modulu PredictReModule. Tato metoda vrací zkratku identifikující zvolený algoritmus získanou z DMSL a lze tedy vždy rozlišit části implementace kódu pro SVM i GLM algoritmus.

V průběhu implementace byl také kladen důraz na udržení konzistence DMSL záznamu v závislosti na zvoleném algoritmu a stavu procesu dolování.

7.3.1 Struktura rozšířeného modulu

Hlavní třída modulu implementující abstraktní třídu MiningPiece je třída PredictReModule. Třídy PredictReModuleInputPanel a PredictReModuleInputPanelGLM implementují panel pro zadávání parametrů podle zvoleného algoritmu. Třída PredictReModuleMineTask a PredictReModuleMineTaskGLM implementují trénovací a testovací fázi tvorby regresního modulu, každá pro jeden typ algoritmu. Třída PredictReModuleOutputPanel implementuje jednotné zobrazení výsledků dolování a zpřístupňuje aplikační fázi přes PredictReModuleMineApplyTask a PredictReModuleMineApplyTaskGLM pro oba algoritmy současně.



Obrázek 7.4 Zjednodušený přehled tříd regresního modulu

Třídy jsou zobrazeny na obrázku 7.4. Oranžově jsou zvýrazněny rozšiřující nové třídy. Zbylé jsou upraveny z původních dle potřeb implementace nového algoritmu.

7.3.2 Vložení komponenty modulu

Po vložení komponenty Regrese do editoru dolovacího procesu dojde ve funkci *afterAcceptEvent()* ke vložení pouze základního elementu DataMiningTask bez upřesnění použitého algoritmu a jeho parametrů. Ty jsou doplněny až po výběru algoritmu pomocí panelu parametrů.

Atributy elementů reprezentující jednotlivé fáze dolování obsažené v elementu Knowledge jsou předvyplněny hodnotami *null*.

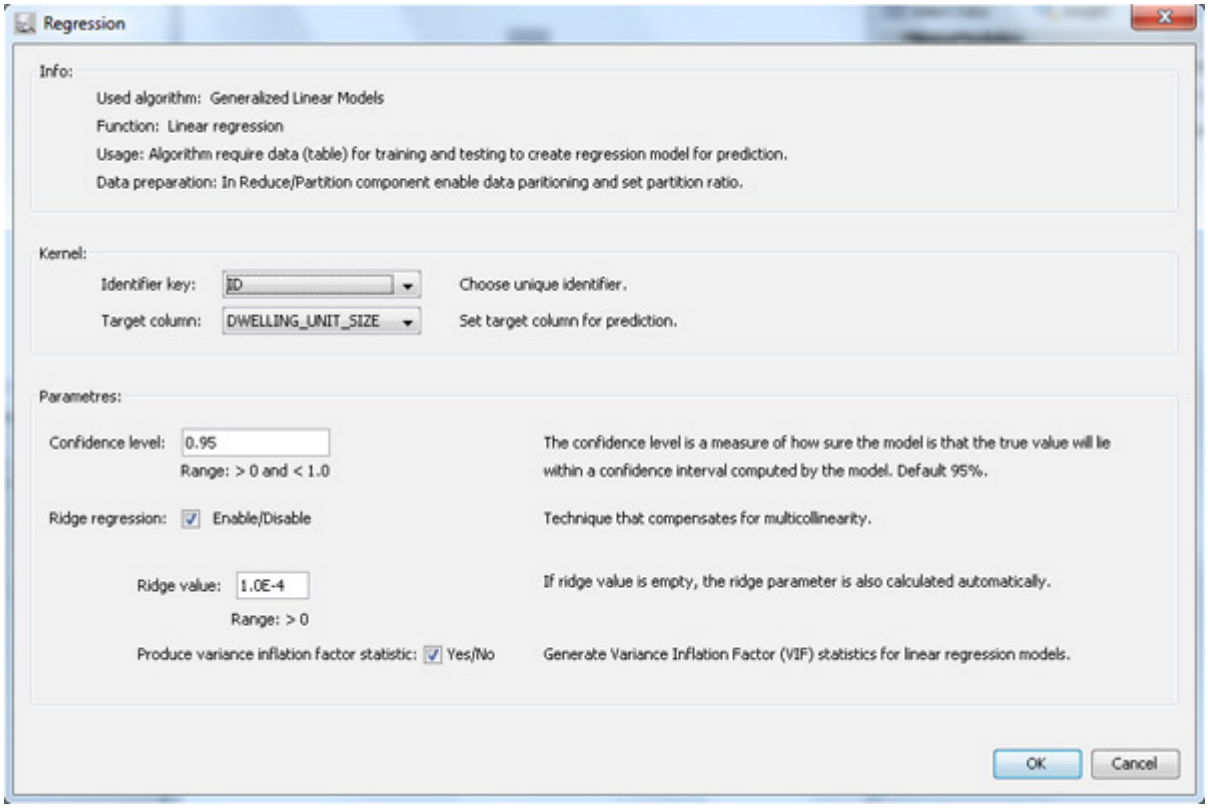
```
<DataMiningTask  language="XML" name="DM537_PredictReModule1" type="Regression">
 <Regression  algorithm="">
  <Usematrix  matrixRef="null"></Usematrix>
  <InputDataType  tableKey="null" target="null"></InputDataType>
  <Parametres ></Parametres>
 </Regression>
</DataMiningTask>

<Knowledge  language="XML" name="DM537_PredictReModule1" type="Regression">
 <BuildTask  end="null" input="null" model="null" name="null" start="null" status="null"></BuildTask>
 <TestTask  end="null" input="null" model="null" name="null" output="null" start="null" status="null">
 </TestTask>
 <ApplyTask  end="null" input="null" model="null" name="null" output="null" start="null" status="null">
 </ApplyTask>
</Knowledge>
```

Obrázek 7.5 Základní stav DMSL po vložení komponenty do editoru dolovacího procesu

7.3.3 Panel parametrů

Zadávání parametrů obou algoritmů je řešeno v metodě *openCustomizingDialog()* třídy PredictReModule.



Obrázek 7.5 Panel pro zadávání parametrů algoritmu GLM

Nejprve je zobrazen panel pro výběr algoritmu. Pokud již byl nějaký dříve vybrán, je nastaven jako výchozí. Po výběru algoritmu je teprve zobrazen panel pro zadání parametrů (obr. 7.5). V případě algoritmu GLM je implementovaný třídou PredictReModuleInputPanelGLM.

V případě změny algoritmu jsou změněny příslušné hodnoty elementu DataMiningTask v DMSL dokumentu. Zaznamenané hodnoty v elementu Knowledge jsou pak vymazány. Tím je vždy zajištěna konzistence záznamů v DMSL pro vybraný algoritmus.

Panel pro zadávání parametrů algoritmu GLM je rozdělen do třech částí. První část má čistě informativní charakter. Ve druhé části je možné nastavit primární klíč vstupní tabulky a definovat predikovaný sloupec tabulky. Nelze nastavit oba parametry na stejný sloupec. Třetí část umožňuje nastavení všech parametrů algoritmu GLM popsaných v kapitole 7.1.1. Při zavření panelu je prováděna kontrola správnosti zadaných hodnot.

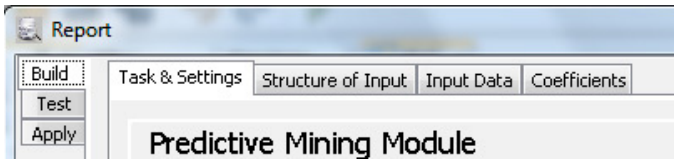
7.3.4 Spuštění dolování

Metoda *run()* spustí první dvě fáze dolování. V první fázi je vytvořen regresní model ze vstupních trénovacích dat a ve druhé je tento model otestován pomocí testovacích dat. Obě tabulky jsou definovány v komponentě Reduce/Partition. Pomocí metody *getRegressionAlgorithm()* je zvolena třída implementující predikční úlohu právě zvoleného algoritmu.

Po vytvoření a otestování modelu jsou příslušně modifikovány elementy v DMSL.

7.3.5 Prezentace výsledků

Prezentace výsledků vytvoření predikčního modelu a výsledků testování modelu je pro oba algoritmy implementována třídou PredictReModuleOutputPanel, která zpřístupňuje i aplikaci modelu na nová data. Panel s výsledky je přístupný přes komponentu Report a je rozdělen záložkami do třech částí, každá reprezentující jednu fázi práce s predikčním modelem (vytvoření, testování, aplikace).



Obrázek 7.6 Záložky panelu pro prezentaci výsledků dolování

Při zobrazení panelu bylo zapotřebí upravit vypisování parametrů dle zvoleného algoritmu a opravit chyby uváznutí vznikající v původní implementaci při plnění a práci s tabulkami. Dále byly reimplementovány exporty tabulek a přepracován grafický vzhled záhlaví záložek.

Detailní implementace třídy PredictReModuleOutputPanel je součástí práce pana Havlíčka [15]. Jednotlivé záložky, práce s výsledky a aplikační fáze jsou popsány v kapitole 8.1.

8 Příklady použití modulů

Tato kapitola je věnována praktickému použití implementovaných modulů a způsobu prezentace vydolovaných znalostí. V jednotlivých krocích je demonstrována detekce odlehlých hodnot a predikce pomocí GLM.

Prvním krokem je v každém případě spuštění aplikace a vytvoření nového dolovacího projektu pomocí nabídky File - New Project. Tímto postupem je spuštěn průvodce vytvořením nového projektu, ve kterém je zapotřebí zvolit možnost Data Mining Project a tlačítkem Next se přesunout na další stránku pro zadání názvu projektu a volby jeho umístění. Opětovným použitím tlačítka Next je zobrazeno poslední okno průvodce pro zadání parametrů připojení k databázi, kde je zapotřebí specifikovat přihlašovací jméno, heslo a url připojované databáze. Po vyplnění všech potřebných údajů je vhodné otestovat připojení k databázi pomocí tlačítka Test connection. Dále je použitím tlačítka Finish ukončen průvodce a vytvořen nový prázdný projekt. Poklepáním na soubor dmsl.xml se otevře plocha editoru a panel s paletou komponent pro definici a modelování dolovacího procesu.

V následujících kapitolách jsou popsány další kroky dolování pro zvolený modul.

8.1 Modul predikce - GLM

Práce s modulem spočívá v nadefinování komponent v grafu dolovacího procesu, nastavení parametrů predikčního algoritmu GLM a zobrazení výsledků dolování přes komponentu Report. Dále může následovat aplikační fáze aplikující regresní model na nová data.

8.1.1 Nadefinování komponent v grafu dolovacího procesu

Vytvoření grafu dolovacího procesu probíhá přetažením komponent z palety na plochu editoru a jejich propojení šipkami. Propojení lze provést při současném stisku levého tlačítka myši, klávesy Ctrl a tažením spojovací šipky mezi komponentami.

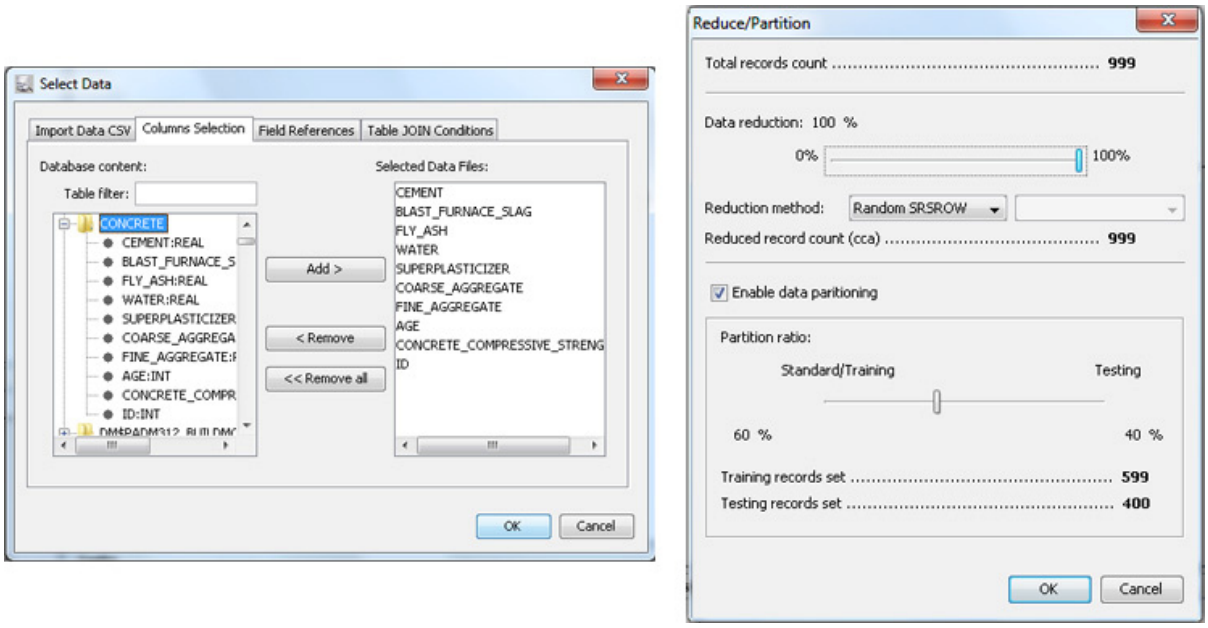
Pro účely dolovací úlohy predikce jsou zapotřebí komponenty Select Data (výběr vstupních dat), Reduce/Partition (rozdělení vstupních dat na trénovací a testovací část), Regression (dolovací komponenta) a Report (zobrazení výsledků a aplikační fáze). Propojení je ukázáno na obrázku 8.1.

Obrázek 8.1 Propojení komponent do dolovacího procesu

8.1.2 Vstupní data, parametry a vytvoření modelu

Pro definici vstupních dat slouží komponenta Select Data, kde po jejím otevření v záložce Columns Selection můžeme v levé části vybrat zdrojovou tabulku a následně pak importovat zvolené sloupce do pravé části, jak je zobrazeno na obrázku 8.2 vlevo. Po výběru tabulky a sloupců je zapotřebí nad komponentou zavolat funkci Run, aby byla vytvořena tabulka vstupních dat.

Vstupní data je zapotřebí rozdělit na dvě části, na trénovací data a testovací. To je možné provést v komponentě Reduce/Partition, kde lze zvolit poměr velikostí obou částí (obr. 8.2 vpravo). Po zavření panelu je potřeba zavolat funkci Run, aby bylo uskutečněno zvolené rozdělení dat.



Obrázek 8.2 Komponenta Select Data a Reduce/Partition

Komponenta Regression nabízí možnost Open, po jejímž spuštění je zobrazen dotaz na výběr dolovacího algoritmu. Po zvolení algoritmu GLM je zobrazen panel pro zadávání parametrů (obr. 7.5). Zde je nutné nastavit primární klíč vstupní tabulky a název predikovaného sloupce. Sloupce nesmí být shodné. Možnosti nastavení parametrů jsou popsány v kapitole 7.1.1.

Vytvoření modelu proběhne po zavolání funkce Run nad komponentou Regression. V případě neúspěchu při vytváření modelu je zobrazeno hlášení s možným řešením. Nejčastěji je zdrojem neúspěchu malé množství dat pro tvorbu modelu nebo nemožnost splnění nastavených parametrů.

8.1.3 Prezentace výsledků

Prezentace výsledků je přístupná přes komponentu Report. Ta obsahuje tři základní záložky odpovídající fázím dolování.

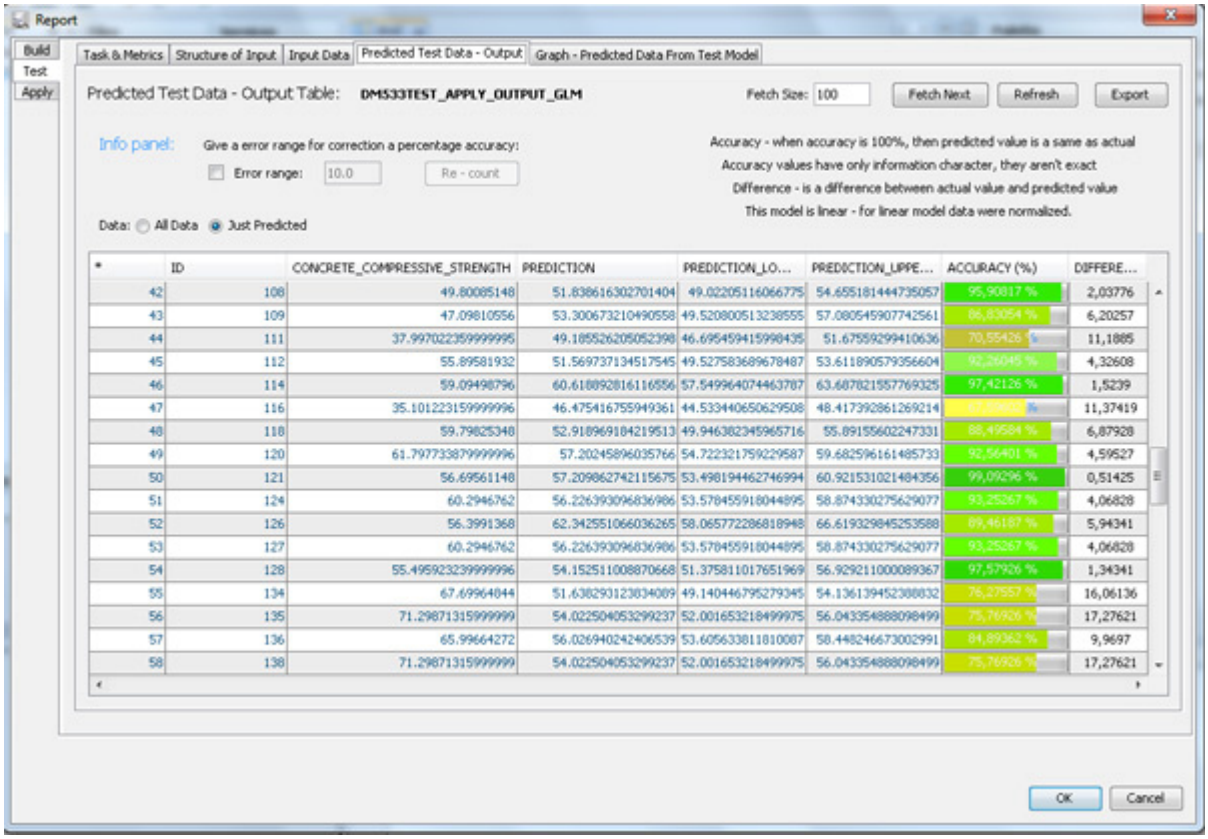
Záložka Build shrnuje fázi vytváření modelu. První list této záložky Task & Settings obsahuje obecné informace o modelu a všechny použité nebo vypočítané parametry při jeho vytváření. Další

listy Structure of Input a Input data obsahují popis a hodnoty vstupních dat. V posledním listu záložky Coefficient jsou zobrazeny koeficienty lineární regrese a Variance inflation factor, pokud bylo požadováno jeho vypočítání a byl vypočítán pro jednotlivé sloupce vstupní tabulky.

Záložka Test shrnuje výsledky fáze testování modelu. V prvním listu Test & Metrics jsou zobrazeny výsledky zjišťovaných metrik udávající kvalitu regresního modelu:

- Root Mean Squared Error (RMSE) – odmocnina z průměru čtverce chyby.
- Mean Absolute Error (MAE) – absolutní střední chyba.
- Mean Actual Value (MAV) – průměr všech aktuálních hodnot.
- Mean Predicted Value (MPV) – průměr všech predikovaných hodnot.

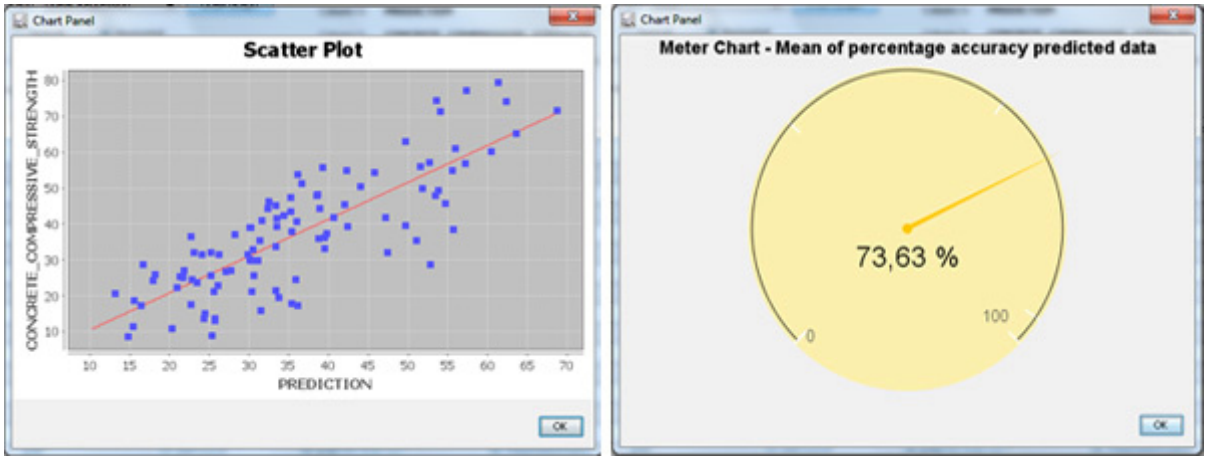
Listy Structure of Input a Input Data obsahují popis a hodnoty testovacích dat. List Predicted Test Data - Output zobrazuje porovnání predikovaného atributu se skutečným a jejich porovnání ve sloupci ACCURACY zobrazující procentuální přesnost predikce. Pokud není zaškrtnuto pole Error range, je výpočet procentuelní přesnosti (Accuracy) přímo závislý na hodnotě aktuální a predikované. V případě, že pole je zaškrtnuto, bere se v potaz korekční hodnota, pro kterou jsme ochotni akceptovat odchylku predikce (obr. 8.3).



Obrázek 8.3 Komponenta Report - porovnání predikované a aktuální hodnoty.

Poslední list Grafh Predicted Data From Test Model slouží k vytváření různých typů grafů popisujících vlastnosti modelu a úspěšnost predikce dat. Nejzajímavější je grafické porovnání

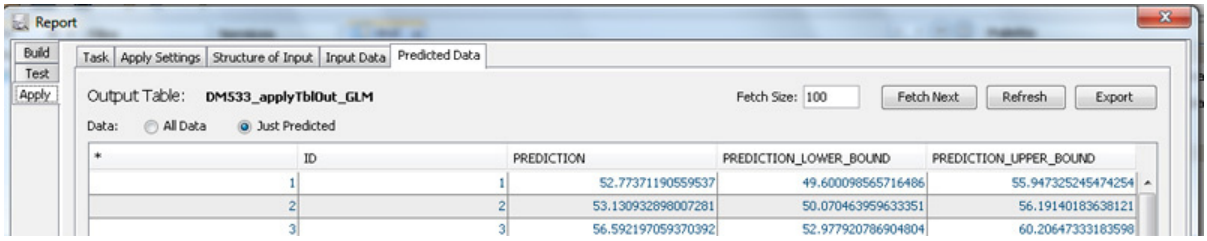
predikované a skutečné hodnoty, které by mělo být v ideálním případě naznačenou přímkou a graf průměrné procentuální úspěšnosti všech predikovaných hodnot (obr. 8.4).



Obrázek 8.4 Graf srovnání predikované a aktuální hodnoty a graf procentuální úspěšnosti predikce

8.1.4 Aplikační fáze

Poslední záložkou panelu komponenty Report je záložka Apply. Pomocí listu Apply settings lze vybrat buď již existující tabulku pro aplikaci regresního modelu, nebo vytvořit novou s ukázkovými daty převzatými z trénovacích dat a možností vložení vlastních hodnot. Poté stačí v listu Input Data pomocí tlačítka Run Apply Task aplikovat regresní model na vybraná data. Pomocí listu Predicted Data lze pak prohlížet výsledné predikované hodnoty. Při vypnuté možnosti ridge regression je zobrazen také horní a dolní limit predikce dle zvolené hodnoty Confidence level, jak lze vidět na obrázku 8.5. Veškeré tabulky v panelu komponenty report lze exportovat do csv souboru nebo databázové tabulky pomocí tlačítka export.



Obrázek 8.5 Výsledky aplikace modelu na nová data

8.2 Modul detekce odlehlých hodnot

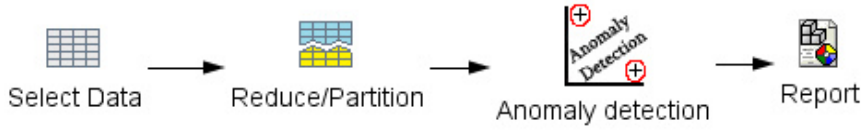
Práce s modulem opět začíná jeho začleněním do dolovacího procesu s ostatními komponentami. Pokračuje nastavením parametrů detekce odlehlých hodnot a vytvořením klasifikačního modelu ze vstupní tabulky obsahující pouze relevantní hodnoty. Vzhledem k nepřítomnosti odlehlých hodnot

v trénovacích datech není možné model testovat, proto je tato část vypuštěna. Získaný model se pak přímo aplikuje pomocí komponenty Report záložky Apply na data s předpokládanou možností výskytu odlehlých hodnot s následnou možností grafického a tabulkového zobrazení detekovaných odlehlých hodnot.

8.2.1 Nadefinování komponent dolovacího procesu

Vytvoření grafu dolovacího procesu probíhá přetažením komponent z palety na plochu editoru a jejich propojení šipkami. Propojení lze provést při současném stisku levého tlačítka myši, klávesy Ctrl a tažením spojovací šipky mezi komponentami.

Pro účely dolovací úlohy detekce odlehlých hodnot jsou zapotřebí komponenty Select Data (výběr vstupních dat), Reduce/Partition (omezení počtu vstupních dat), Anomaly Detection (dolovací modul) a Report (zobrazení výsledků a aplikační fáze). Propojení je ukázáno na obrázku 8.6.



Obrázek 8.6 Propojení komponent do dolovacího procesu

8.2.2 Vstupní data, parametry a vytvoření modelu

Pro definici vstupních dat slouží komponenta Select Data, kde po jejím otevření v záložce Columns Selection můžeme v levé části vybrat zdrojovou tabulku a následně pak importovat zvolené sloupce do pravé části, jak je zobrazeno na obrázku 8.2 vlevo. Vstupní tabulka pro vytvoření klasifikačního modelu detekce odlehlých hodnot musí obsahovat pouze relevantní data. Po výběru tabulky a sloupců je zapotřebí nad komponentou zavolat funkci Run. Tím je vytvořena tabulka vstupních dat.

Vstupní data je možné omezit v počtu řádků. To je možné provést v komponentě Reduce/Partition, zde také zrušíme volbu Enable data partitioning. Rozdělení tabulek není pro tento modul zapotřebí (obr. 8.2 vpravo).

Komponenta Anomaly Detection zpřístupňuje panel pro zadávání parametrů one-class SVM algoritmu přes možnost Open. Zde je zapotřebí definovat primární klíč vstupní tabulky a parametry algoritmu. Obrázek 6.9 zobrazuje panel a možnosti nastavení jednotlivých parametrů jsou popsány v kapitole 6.2.1.

Po nastavení parametrů je zavoláním funkce Run nad komponentou Anomaly Detection vytvořen klasifikační model.

8.2.3 Prezentace výsledků

Prezentace výsledků je přístupná přes komponentu Report. Ta obsahuje dvě záložky odpovídající fázím dolování.

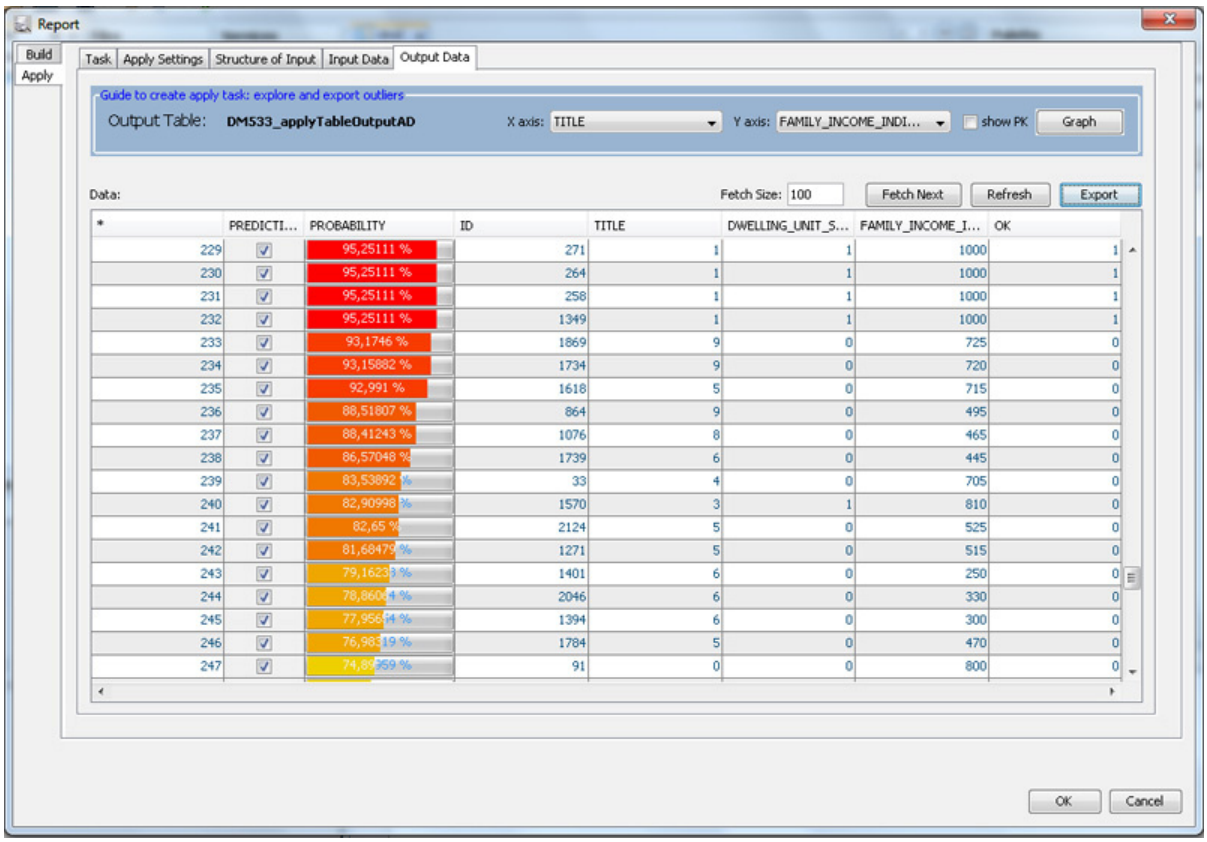
Záložka Build shrnuje fázi vytváření modelu. První list Task & Settings obsahuje obecné informace o modelu a všechny použité nebo vypočítané parametry při jeho vytváření. Další listy Structure of Input a Input data obsahují popis a hodnoty vstupních dat.

Záložka Apply obsahuje nejstěžejnější část vytváření a zobrazení výsledků detekce odlehlých hodnot a je jí věnována následující kapitola.

8.2.4 Aplikační fáze

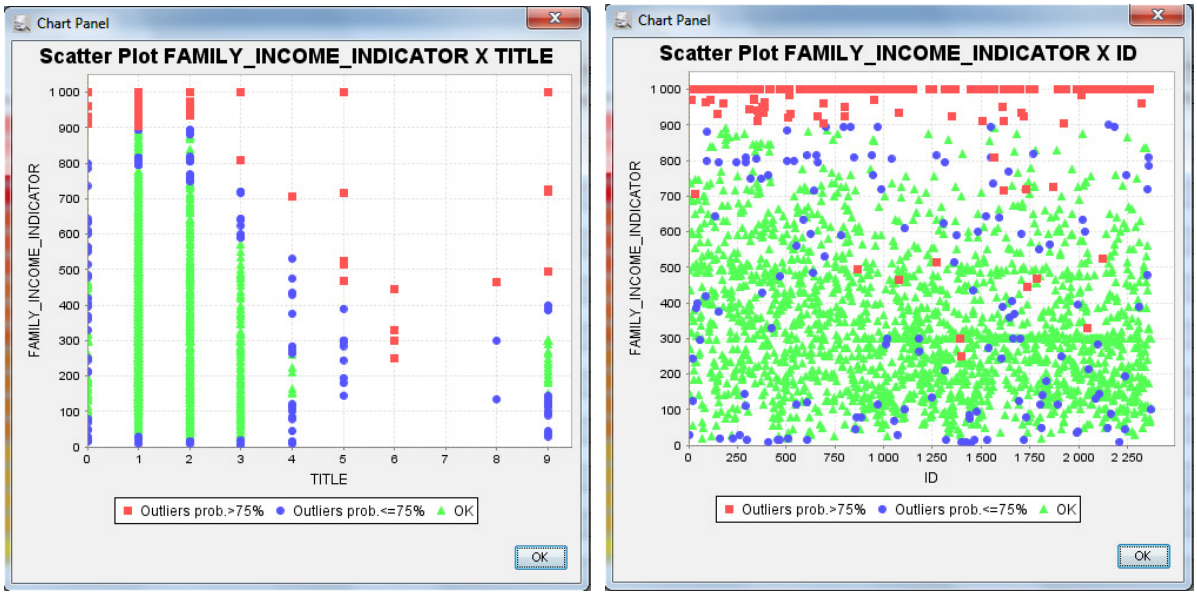
Pomocí listu Apply Settings v záložce Apply lze vybrat tabulku z databáze s předpokladem výskytu odlehlých hodnot nebo nechat vygenerovat tabulku novou s náhodnými ukázkovými hodnotami ze vstupní tabulky. Po výběru tabulky jsou v dalším kroku zobrazena data k aplikaci modelu v listu Input Data a v listu Structure of Input je jejich popis. V případě vytvořené nové tabulky je možnost vložená ukázková data měnit. Samotná aplikace klasifikačního modelu je spuštěna pomocí tlačítka Run Apply Task.

Po aplikaci modelu na data je zpřístupněn list Output Data, ve kterém jsou v tabulce zobrazeny nalezené odlehlé hodnoty s uvedenou mírou pravděpodobnosti náležitosti do třídy odlehlých hodnot ve sloupci PROBABILITY (obr. 8.7).



Obrázek 8.7 Panel s výsledky detekce odlehlých hodnot

Nad celou tabulkou je možné tvořit grafy ukazující rozložení atributů v závislosti na jiném atributu nebo primárním klíči tabulky. Z těchto grafů se dá odhadnout rozložení množiny relevantních dat a určit tak, jestli detekovaná hodnota je skutečně odlehlá. V grafu jsou pro přehlednost hodnoty považované za relevantní označeny zeleně, hodnoty s pravděpodobností náležitosti mezi odlehlé 75 % a méně jsou označeny modře a hodnoty s vyšší pravděpodobností jsou označeny červeně, jak je vidět na obrázku 8.8.



Obrázek 8.8 Panel s výsledky detekce odlehlých hodnot

Vydolované výsledky je možné exportovat do souboru csv nebo databázové tabulky a je možné si zvolit buď pouze vybrané řádky s odlehlými hodnotami, nebo pouze relevantní řádky, popřípadě je možné vyexportovat celou tabulku se všemi hodnotami.

9 Závěr

Celkovým cílem práce je navržení a implementace nových dolovacích modulů pro systém vyvíjený na platformě NetBeans. K dosažení tohoto cíle bylo nutné se v první fázi seznámit s problematikou získávání znalostí z databází a seznámit se důkladně s aktuální implementací jádra a s již existujícími dolovacími moduly aplikace Dataminer.

Z takto získaného přehledu o možnostech dolování dat a již implementovaných dolovacích metodách bylo možné určit cíle práce. Po konzultaci s vedoucím práce bylo zvoleno, že prvním cílem práce bude rozšíření již existujícího modulu regrese o nový dolovací algoritmus vícenásobné lineární regrese založený na zobecněných lineárních modelech.

Při zkoumání dalších možností dolování dat bylo zjištěno, že není pokryta možnost pro dolování odlehlých hodnot. Proto bylo rozhodnuto, že dalším cílem práce bude implementace a integrace této dolovací metody do dolovacího systému.

Obě tato rozšíření využívají podporu dolování dat na straně databázového serveru Oracle data mining. Před samotnou implementací bylo tedy zapotřebí prozkoumat možnosti práce s ODM v kontextu implementace nových dolovacích algoritmů pro nové moduly aplikace Dataminer.

Při samotné implementaci bylo zapotřebí řešit začlenění nových řešení do celého systému aplikace Dataminer a implementovat možnosti grafické reprezentace vydolovaných znalostí.

V současném stavu je tedy možné využít pro predikci v aplikaci Dataminer regresní modul s plnohodnotnou implementací dvou regresních algoritmů SVM a GLM. Přičemž součástí implementace je i možnost zobrazení vydolovaných znalostí a jejich znázornění v grafech. Pomocí implementace nového modulu pro dolování odlehlých hodnot je také umožněna detekce odlehlých hodnot s možností jejich zkoumání, zobrazení, případně odstranění.

Literatura

[1] ZENDULKA, J., BARTÍK, V., LUKÁŠ, R. aj. *Získávání znalostí z databází – studijní opora.* Brno: FIT VUT v Brně, 2006.

[2] HAN, J. a KAMBER, M. *Data Mining: Concepts and Techniques.* 2. vyd. Elsevier Inc., 2006. 770 s. ISBN 978-1-55860-901-3.

[3] *Regresní analýza, Wikipedia* [online], 2009. [citace: 16. 12. 2009]
URL http://cs.wikipedia.org/wiki/Regresn%C3%AD_anal%C3%BDza

[4] LITSCHMANNOVÁ, M. *Regrese* [online], 2008. [citace: 16. 12. 2009]
URL http://www.am.vsb.cz/litschmannova/STA1/Cviceni/PDF/14cRegrese.PDF

[5] KÁBA, B. *Identifikace odlehlých pozorování ve statistických datech*
[online], 2001. [citace: 16. 12. 2009]
URL www.agris.cz/etc/textforwarder.php?iType=2&iId=125823

[6] *Typy grafů v R, Numerická simulace* [citace: 20. 12. 2009], 2008.
URL http://wood.mendelu.cz/cz/sections/FEM/?q=node/82

[7] SACHS, L. *Applied Statistics*. New York: Springer Verlag, 1984.

[8] *Datamining, hrabák.org* [online], 2009. [citace: 20. 12. 2009]
URL http://www.hrabak.org/datamining.htm

[9] KOTÁSEK, P. *DMSL: Data Mining Specification Language*: *disertační práce*. Brno: FIT VUT v Brně, 2003.

[10] *NetBeans, Wikipedia* [online], 2009. [citace: 20. 12. 2009]
URL http://cs.wikipedia.org/wiki/NetBeans

[11] *JSR-73 Java Documentation, Oracle* [online], 2008. [citace: 20. 12. 2009]
URL http://www.oracle.com/technology/products/bi/odm/jsr-73/index.html

[12] *Support vector machines (SVM)* [online], 2007. [citace: 29. 12. 2009]
URL http://is.muni.cz/el/1433/podzim2006/PA034/09_SVM.pdf?fakulta=1433;obdobi=3523;kod=PA034

[13] ŠEBEK, M. *Rozšíření funkcionality systému pro dolování z dat na platformě NetBeans: diplomová práce*. Brno: FIT VUT v Brně, 2009.

[14] HENKL, T. *Dolovací moduly systému pro dolování z dat na platformě NetBeans: diplomová práce*. Brno: FIT VUT v Brně, 2009.

[15] HAVLÍČEK, D. *Vytvoření nových predikčních modulů v systému pro dolování z dat na platformě NetBeans: diplomová práce*. Brno: FIT VUT v Brně, 2009.

[16] VESELÝ, M. *Aplikace GLM modelu v provozní praxi: diplomová práce*. Brno: PF Masarykova univerzita, 2007.

[17] ANDĚL, J. *Matematická statistika*. Praha: SNTL/Alfa, 1978.

[18] *Klasický lineární model* [online], 2006. [citace: 10. 4. 2010]
URL http://iastat.vse.cz/regrese/Regrese5.htm

[19] MADER, P. *Dolovací moduly systému pro dolování z dat v prostředí Oracle: diplomová práce.* Brno: FIT VUT v Brně, 2009.

[20] KMOŠČÁK, O. *Vytvoření nových klasifikačních modulů v systému pro dolování z dat na platformě NetBeans: diplomová práce*. Brno: FIT VUT v Brně, 2009.

[21] *The Data Mining Sample Programs* [online], 2008. [citace: 15. 4. 2010]
URL http://download.oracle.com/docs/cd/E11882_01/datamine.112/e12217/sampleprogs.htm#CHDDFBEC

[22] *The Data Mining Java API* [online], 2008. [citace: 19. 4. 2010]
URL http://download.oracle.com/docs/cd/B28359_01/datamine.111/b28131/java_api.htm

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

Seznam příloh

Příloha A: Obsah přiloženého CD

Příloha A: Obsah přiloženého CD

Adresářová struktura na přiloženém CD s informativním popisem jejich obsahu:

- **Dataminer**
  - **build** – adresář obsahující distribuční spustitelnou verzi Datamineru.
  - **scr** – adresář obsahující zdrojové kódy aplikace Dataminer včetně modulů detekce odlehlých hodnot a regrese.
- **Examples** – adresář obsahuje obrázkový tutorial k použití modulu regrese a detekce odlehlých hodnot a soubor s testovacími daty.
- **Thesis** – adresář obsahuje text diplomové práce.