VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

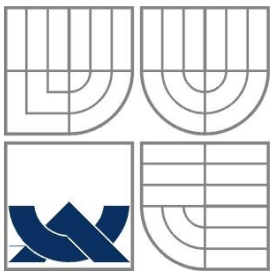
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

KLASIFIKAČNÍ DOLOVACÍ MODULY SYSTÉMU PRO DOLOVÁNÍ Z DAT NA PLATFORMĚ NETBEANS

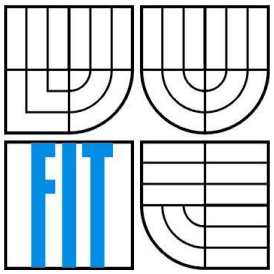
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE NORBERT KUZMA
AUTHOR

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

KLASIFIKAČNÍ DOLOVACÍ MODULY SYSTÉMU PRO DOLOVÁNÍ Z DAT NA PLATFORMĚ NETBEANS

CLASSIFICATION MINING MODULES OF DATA MINING SYSTEM ON NETBEANS PLATFORM

BAKALÁŘSKÁ PRÁCE
BACHELOR‘S THESIS

AUTOR PRÁCE — NORBERT KUZMA
AUTHOR

VEDOUCÍ PRÁCE — Ing. MICHAL ŠEBEK
SUPERVISOR

BRNO 2010

Abstrakt

Tato bakalářská práce se zabývá dolováním z dat a vytvořením dolovacího modulu pro systém pro dolování z dat, který je vyvíjen na FIT. Jedná se o klientskou aplikaci skládající se z jádra a jeho grafického uživatelského rozhraní a nezávislých dolovacích modulů. Aplikace je implementována v jazyce Java a její grafické uživatelské rozhraní je postaveno na platformě NetBeans.

Obsahem této práce bude uvedení do problematiky získávání znalostí z databází a následně seznámení s neuronovými síťemi, pro které bude následně implementován samotný dolovací modul. Dále bude popsána implementace tohoto modulu.

Abstract

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

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

Klíčová slova

Získávání znalostí z databází, dolování z dat, modul, klasifikace, backpropagation, neuronové sítě, Java, NetBeans.

Keywords

Knowledge discovery in databases, data mining, module, classification, neural network, backpropagation, Java, NetBeans

Citace

Kuzma Norbert: Klasifikační dolovací moduly systému pro dolování z dat na platformě NetBeans, bakalářská práce, Brno, FIT VUT v Brně 2010

Klasifikační dolovací moduly systému pro dolování z dat na platformě NetBeans

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Michala Šebka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Norbert Kuzma
14.15.2010

Poděkování

Chcel by som poďakovať Ing. Michalovi Šebkovi za jeho podporu, rady, ochotu, trpezlivosť a ústretovosť pri konzultáciách k tejto bakalárskej práci.

Obsah

1 Úvod

V posledných rokoch sa v databázach ukladá obrovské množstvo dát. Ich množstvo bolo ešte nedávno nepredstaviteľné, veď každá banková transakcia, kúpa lístka na vlak alebo pracovná cesta sú uložené v databázach. S pribúdajúcim množstvom dát vznikla myšlienka využiť tieto dáta aj na niečo iné ako ukladanie prvoplánových informácií.

Jedným z prvých spôsobov boli dátové sklady a s nimi technológia *OLAP* (z angl. *On Line Analytical Processing*), v ktorých boli dáta použité pre analýzu oddelené od dát z pôvodnej databáze. OLAP nástroje sú používané v interakcii s používateľom a postupom času vznikla požiadavka tento proces automatizovať. Tak vznikla myšlienka a pojem *získavania znalostí z databází* (*angl. Knowledge Discovery in databases* - KDD) zaužívaným pomenovaním je aj *dolovanie z dát* (angl. *Data mining*). Dolovanie z dát je zložitý proces a preto sú za týmto účelom vyvíjané nástroje, ktoré ľuďom uľahčujú a zjednodušujú celý proces. Na Fakulte informačných technológií Vysokého učení technického v Brne sa vyvíja systém, ktorý má študentom umožniť odskúšať si dolovanie z dát v praxi. Tento systém je vyvíjaný na v jazyku JAVA na platforme Netbeans.

V mojej bakalárskej práci sa budem zaoberať rozšírením vyvíjaného systému o klasifikačný modul založený na neurónových siteťach konkrétne sieť so *spätným šírením chyby* (angl. *backpropagation*). Práca nadväzuje a čerpá z projektov riešených v minulých rokoch, ktoré sa systému pre dolovanie z dát venovali od pánov Ing. Krásného a Ing. Šebka.

V druhej kapitole sa budem venovať problematike získavania znalostí z databáz všeobecne a konkrétnejšie rozoberiem najmä klasifikáciu. V tretej kapitole sa budem venovať neurónovým sieťam a to hlavne sieti backpropagation a jej algoritmu. Štvrtá kapitola sa venuje použitým technológiám v rámci aplikácie Dataminer ako aj popisu jej základných častí. Piata kapitola sa venuje implementácii modulu a príklad jeho použitia v aplikácii je uvedený v kapitole šesť. V siedmej kapitole sa zameriam na testovanie modulu z pohľadu vplyvu vybraných parametrov na proces dolovania. V závere zhodnotím prínos tejto práce ako aj jej možné rozšírenia.

2 Získavanie znalostí z databáz

Získavanie znalostí z databáz ako pojem vyjadruje extrahovanie pre nás zaujímavých a užitočných modelov a vzorov dát. Zaujímavosťou a užitočnosťou sa rozumie, že dáta, ktoré získame pomocou dolovania, sú netriviálne, nedajú sa získať SQL dotazom, sú pre nás skryté. Tieto modely a vzory reprezentujú znalosti získané z dát. Môžeme si to predstaviť tak, že keď človek nakupuje v obchode určitý druh tovaru (napr. PC), kúpi si k nemu zvyčajne aj iný druh tovaru (napr. klávesnicu, monitor, myš).

Toto má využitie v marketingu, keď môže obchodník nalákať zákazníka na nízku cenu PC a predať mu k tomu v odporúčanej zostave mierne predražené ostatné komponenty. Tento príklad je veľmi triviálny, avšak podstatu užitočnosti dát získaných dolovaním vykresľuje. Databáza uchováva dáta o nákupoch zákazníkov a až dolovanie z týchto dát nám poskytne modely a vzory častých nákupov. Proces získavania znalostí z databáz nie je jednoduchý a jeho časti opíšem v nasledujúcich podkapitolách. Viac sa o dolovaní z dát možno dozvedieť z [2].

2.1 Získavanie znalostí z databáz ako proces

Dolovanie z dát je ucelený proces, kde samotné dolovanie je len jednou z častí tohto procesu, ktorého časti charakterizujem v tejto podkapitole.

1. **Čistenie dát** - odstránenie chybných a zašumených dát,
2. **integrácia dát** - ak je zdrojom pre dolovanie z dát viac databáz spojíme ich do jedného celku,
3. **výber dát** - z databáze sa vyberú len dáta relevantné pre dolovanie,
4. **transformácia dát** - cieľom je upraviť dáta do konsolidovanej podoby vhodnej pre dolovanie, môže ísť napríklad o sumarizáciu alebo agregáciu atribútov,
5. **dolovanie dát** - jadro celého procesu, ktorého cieľom je aplikáciou určitých algoritmov získať z dát model alebo vzor reprezentujúci dáta,
6. **hodnotenie modelov a vzorov** - výber pre nás zaujímavých a užitočných modelov a vzorov dát,
7. **prezentácia znalostí** - cieľom je prezentovať výsledky dolovania z dát užívateľovi.

Prvé štyri body procesu patria do takzvaného predspracovania dát, ktoré je hlavne pre metódy založené na neurónových sieťach veľmi významné, tento význam predspracovania opíšem bližie v kapitole o neurónových sieťach. V kapitole 2.2 sa budem bližšie venovať procesu predspracovania dát vo všeobecnosti. Viac o pojme získavania znalostí z databáz je uvedené v [2].

2.2 Predspracovanie dát

Predspracovanie dát je dôležitou časťou procesu dolovania dát, pretože upravuje dáta do podoby, ktorá je prijateľná pre dolovací algoritmus a bez tohto procesu by dolovanie z dát bolo nepresné, ak by bolo vôbec možné. Viac o predspracovaní dát je uvedené v [4].

2.2.1 Čistenie dát

Prvou časťou predspracovania dát je ich čistenie, keďže hodnoty atribútov jednotlivých záznamov môžu chýbať alebo byť chybne zadané (zašumené). Čistenie dát má dve fáze a to detekciu nezrovnalostí a opravenie nezrovnalostí.

2.2.2 Integrácia dát

Dáta, z ktorých sa chystáme dolovať obyčajne pochádzajú z viacerých zdrojov a preto ich musíme integrovať do jedného úložiska. Pri integrácii môže dôjsť k nekonzistencii dát. Medzi hlavné problémy riešené pri integrácii dát patria :

Konflikt schémy - keďže dáta pochádzajú z viacerých zdrojov môže byť tá istá informácia reprezentovaná viacerými schémami. Ako príklad môže poslúžiť adresa ktorá môže byť v jednom zdroji reprezentovaná trojicou atribútov mesto, ulica, PSČ a v inom zdroji len jedným atribútom adresa. Inými slovami sa to dá popísať že je nutné integrovať metadáta z viacerých zdrojov do jedných metadát popisujúcich výsledný zdroj metadát.

Konflikt hodnôt - ide o prípad, keď hodnoty odpovedajúcich si atribútov majú rôznu reprezentáciu. To môže byť spôsobené napríklad rôznymi konvenciami, stupnicami, jednotkami a podobne.

Konflikt identifikácie - dáta môžu byť v databázach prezentované rôzne a tak môže vzniknúť situácia že ten istý objekt je v databázach pod rôznymi identifikátormi, alebo naopak že ten istý identifikátor identifikuje viac rôznych objektov.

Odstránenie redundancie - v databázach sa môžeme stretnúť s tým, že niektoré atribúty sa dajú ľahko odvodiť od iných atribútov. Ak je napríklad v databáze údaj o dátume narodenia dá sa z neho ľahko odvodiť vek a tak vzniká redundancia, ktorej sa chceme pred dolovaním z dát zbaviť, keďže veľa takýchto atribútov môže spôsobiť nezmyselné výsledky dolovacieho procesu.

2.2.3 Transformácia dát

Vyhladenie - ide o odstránenie šumu v dátach.

Normalizácia - transformácia do intervalov prijateľnejších pre dolovanie. Väčšinou <-1,1> ,alebo <0,1>.

Agregácia - detailné dáta sa agregujú cez jednu alebo viac dimenzií. Pod dimenziou sa rozumie napr. čas, pobočka, prepravca, ...

Generalizácia - ide v podstate o zovšeobecnenie dát, ktoré môžu byť v konceptuálnej hierarchii na nižšej úrovni ako je potrebné pre dolovanie. Hodnoty ulica, alebo mesto môžu byť generalizované na názov kraju alebo štátu.

Konštrukcia atribútov - z jedného, alebo viacerých atribútov môžeme vytvoriť nový atribút prijateľnejší pre dolovanie, napr. ak nás u športovca zaujíma či splnil olympijský limit môžeme z atribútov čas a limit vytvoriť nový atribút, ktorý hovorí či sa športovec kvalifikoval.

2.2.4 Redukcia dát

Dolovanie z dát sa prevádza nad obrovským množstvom dát a cieľom tohto kroku je dáta zredukovať pri zachovaní ich výpovednej hodnoty.

Agregácia dátovej kocky - dáta sa redukujú agregovaním údajov, typické pre dátové sklady.

Výber podmnožiny atribútov - cieľom je z množiny atribútov odstrániť atribúty redundantné, alebo málo relevantné pre výsledok dolovania.

Redukcia dimenzionality - dáta sa zakódujú do takej podoby, že dôjde k redukcii, ale bude s nimi možné prevádzať potrebné informácie. Typickým spôsobom je napr. vlnková transformácia a analýza hlavných komponent.

Redukcia numerozity - jedná sa o redukciu počtu hodnôt, pri ktorej sa dáta zredukujú na model a sú reprezentované jeho parametrami, alebo sú reprezentované v zredukovanej podobe.

Diskretizácia a generácia konceptuálnej hierarchie - hodnoty atribútov sú nahradené intervalmi, alebo pojmami z konceptuálnej hierarchie. Ide o špeciálny prípady kedy sa neredukuje počet n-tíc, ale počet atribútov.

2.3 Typy dolovacích úloh

Keďže existuje viacero druhov znalostí, ktoré chceme dolovacím procesom získať, musí byť aj viacero druhov dolovacích metód, ktoré môžu byť implementované rôznymi algoritmami.

Typy dolovacích úloh sa rozdeľujú na dve základné skupiny. Viac o typoch dolovacích úloh je uvedené v [1].

Deskriptívne - deskriptívna úloha charakterizuje dáta pomocou sumarizácie ich vlastností za účelom ich štúdia. Ako príklad môže poslúžiť už spomínaný nákup PC a k nemu nákup monitoru ich spoločnou vlastnosťou je častý súbežný nákup.

Prediktívne - na základe analýzy dát sa o predpoveď budúceho chovania. Jedná sa o dedukciu napr. toho či bude zákazník banky schopný splácať úver. Medzi prediktívne metódy patrí klasifikácia a predikcia, ktoré bližšie popíšem v ďalšej podkapitole.

2.3.1 Popis konceptu/tried

Dáta môžu byť spájané s určitou triedou, alebo konceptom a potom je možné tieto triedy a koncepty popísať určitým jednoduchým a presným spôsobom. Popis možno vytvárať dvoma spôsobmi:

- **Charakterizácia dát –** je súhrnný popis vlastností triedy. Výsledkom býva charakteristika napr. určitého typu tovaru. Charakterizácia hľadá hodnoty atribútov, ktoré sú podobné s vlastnosťami danej triedy, alebo konceptu.
- **Diskriminácia dát –** nepopisuhe triedu všeobecne, ale snaží sa ju porovnávať s jednou, alebo viacerými inými triedami. Hľadá hodnoty atribútov, ktoré sa čo najviac odlišujú.

2.3.2 Frekventované vzory

Sú to vzory, ktoré sa v dátach často vyskytujú. Tieto úlohy ukazujú zaujímavé spojitosti medzi atribútmi, ktoré sa používajú napríklad pre analýzu nákupného košíku v internetovom ochode. Jedná sa o vyhľadávanie asociácií a korelácií. V asociačnom pravidle nejaký atribút podmieňuje iný atribút. Ako príklad uvediem už spomínaný nákupný košík:

kupuje(X, monitor) => kupuje(X, PC) [podpora = 5%, spoľahlivosť = 30%]

Dôležitými údajmi sú podpora a spoľahlivosť, ktoré určujú dôležitosť zastúpenia frekventovaného vzoru v dátach, ktoré analyzujeme. *Podpora* značí, v koľkých percentách nákupov boli spoločne tieto položky nakúpené. V našom prípade boli monitor spolu s PC zostavou zakúpené v 5% prípadov. *Spoľahlivosť* vyjadruje počet zastúpenia položiek na pravej strane pravidla vzhľadom k počtu položiek na ľavej strane pravidla. V príklade si zákazník kúpil PC zostavu v 30% nákupov, kedy si kúpil aj monitor.

2.3.3 Zhluková analýza

Zhlukovanie rozdeľuje jednotlivé objekty do tried na základe podobnosti. Hľadá také objekty, ktoré sú s čo najviac podobné a zhlukuje ich do tried. Snaží sa zároveň o čo najväčšiu diverzifikáciu jednotlivých tried.

2.3.4 Analýza odľahlých objektov

Použitie odľahlých objektov sa od ostatných dolovacích úloh líši v tom, že sa nehľadajú vzory, ktoré sa vyskytujú často, práve naopak hľadá vzory, ktoré sa od ostatnej vzorky dát niečím líšia takzvané odľahlé objekty. Metóda sa používa napríklad pri hľadaní podozrivých bankových transakcií. Je zrejmé že predspracovanie v podobe normalizácie alebo odstránenia šumu v dátach by u tejto metódy mali negatívny vplyv na výsledky, ktoré by stratili výpovednú hodnotu.

2.3.5 Evolučná analýza

Sleduje chovanie jednotlivých atribútov v čase. Zameriava sa na intervaly, ktoré sú medzi jednotlivými opakovaniami, alebo rýchlosť zmeny hodnoty atribútu v čase. Evolučná analýza sa využíva predovšetký pri odhade ceny komodít na burze.

2.4 Klasifikácia a predikcia

Ako už bolo povedané klasifikácia a predikcia patria medzi prediktívne úlohy, najprv sa budem venovať klasifikácii, potom predikcii a na záver ich porovnám.

2.4.1 Klasifikácia

Cieľom klasifikácie je nájsť taký model, ktorý popisuje a súčasne rozlišuje triedy dát a potom ho použiť pre predikciu triedy, do ktorej bude objekt, ktorého zaradenie nepoznáme patriť [1]. Zjednodušene povedané je to proces, ktorý umožní dáta rozdeliť do daných tried na základe ich vlastností. Klasifikačný proces prebieha v následujúcich krokoch [1]:

1. **Trénovanie** – na základe analýzy tzv. trénovacej množiny je vytvorený klasifikačný model.
2. **Testovanie** – hodnotenie kvality vytvoreného modelu použitím testovacích dát.
3. **Aplikácia** – použitie modelu pre klasifikáciu objektu, ktorého triedu nepoznáme.

V niektorých zdrojoch sa aplikácia neuvádza ako časť klasifikačného procesu a tento sa skladá len z fázy trénovacej a fázy testovacej. Trénovanie prebieha na trénovacej množine dát a testovanie na testovacej množine dát, pričom obe tieto množiny sú dáta, pri ktorých poznáme aj triedu, do ktorej patria. V prípade trénovania sa táto skutočnosť využíva na vytvorenie modelu a v prípade testovania na zistenie pomeru správne a nesprávne priradených tried k dátam, teda k presnosti klasifikačnej metódy. Podľa testovania sa určí úspešnosť klasifikačného modelu a prehlási sa za správny, alebo sa zistí že jeho úspešnosť je nepostačujúca a treba ho upreviť, alebo je treba vytvoriť model nový. Klasifikačný model môže mať rôzne podoby ako napr. matematické formule, rozhodovacie stromy, klasifikačné pravidlá, alebo neurónové siete.

2.4.1.1 Príprava dát pre klasifikáciu

Aby bol proces klasifikácie efektívny a dosahoval čo najlepšie výsledky, je treba špeciálnym spôsobom upraviť surové dáta z databázy. Medzi tieto úpravy patrí [5]:

1. **Čistenie dát** – odstránenie šumu v dátach a nahradenie chýbajúcich hodnôt najčastejšie sa vyskytujúcou hodnotou, či priemernou hodnotou atribútu, prípadne vynechanie neúplných vzorkov dát.

2. **Významnostná analýza** – odstránenie atribútov, ktoré sú pre klasifikáciu zbytočné. S týmto problémom si neurónové siete vedia ľahko poradiť.
3. **Transformácia dát** – nazývaná aj prevod dát. Jedná sa o zovšeobecnenie dát, napríklad prevod čísel na diskrétne hodnoty. Napríklad cena PC zostavy z presnej hodnoty na hodnotu drahá/lacná. Špeciálnym prípadom transformácie je normalizácia, kedy sa všeobecný interval prevádza na interval <0,1>, prípadne interval <-1,1>.

2.4.1.2 Porovnávanie klasifikačných metód

K porovnávaniu klasifikačných metód sa používajú prevažne tieto kritériá:

1. **Presnosť predpovedi** – v koľkých precentách vie model správne klasifikovať dáta z testovacej množiny prípadne nové dáta.
2. **Rýchlosť** – výpočetná zložitosť generovanie a následného používania klasifikačných pravidiel.
3. **Robusnosť** – schpnosť vysporiadať sa s dátovým šumom a chýbajúcimi dátami.
4. **Stabilita** – schopnosť vytvoriť správny model pre veľký objem dát. Toto býva problémom pri neurónových sieťach pretože pri obovskom množstve dát sa môže stať že sa sieť v podstate naučí presne túto množinu dát a dochádza k tzv. preučeniu, kedy sieť nevie správne reagovať na iné dáta ako dáta z trénovacej množiny.
5. **Interpretovateľnosť** – zložitosť daného problému pre jeho pochopenie.

2.4.2 Predikcia

Predikcia je proces, ktorý umožní priraďovať dátam hodnoty, ktoré sú na rozdiel od klasifikácie spojitého charakteru. Príkladom je predikcia vývoju cien komodít na burzových trhoch na základe predošlého správania sa trhu a vývoja cien. Najznámejšou metódou predikcie je tzv. lineárna jednoduchá a lineárna násobná regresia alebo metóda support vector machines (SVM). Mnohé nelineárne problémy sa dajú na tieto typy regresie transformovať.

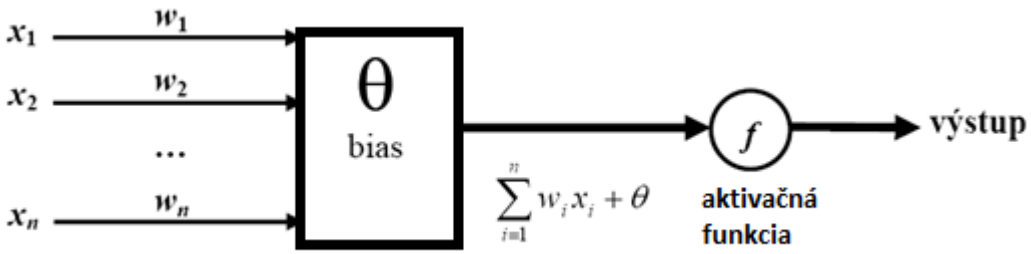
Klasifikácia aj predikcia patria medzi prediktívne dolovacie úlohy, avšak rozdiel medzi nimi je v tom, že klasifikácia sa zaoberá rozdelovaním objektov do tried na základe ich vlastností a predikcia predpovedá hodnotu spravidla spojitého charakteru napríklad už spomenutú cenu komodity na trhu.
Viac o druhoch dolovacích úloh sa možno dozvedieť z [2].

3 Neurónové siete

Neurónové siete v informatike vychádzajú so skutočných neurónových sietí a neurónov, ktoré sú súčasťou aj ľudskej neurónovej sústavy.

3.1 Perceptrón

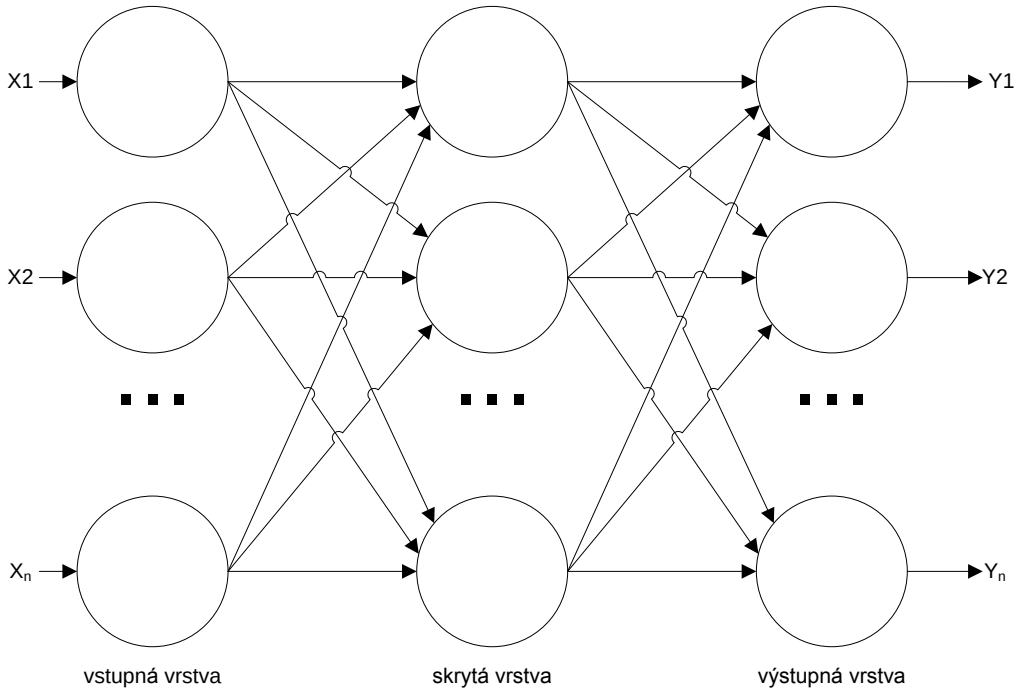
Ľudský neurón je bunka, ktorá má viacero vstupov a jeden jediný výstup a z tohto vychádza aj neurón v informatike [3]. Základným typom neurónu je perceptrón, ktorý bol navrhnutý Frankom Rosenblattom. Ako je na obrázku ukázané perceptroón má viacero vstupov s určitou váhou a jeden výstup, ktorý samozrejme môže smerovať do iných neurónou a tak sa vytvára neurónová sieť. Na obrázku 3.1 je model umelého neurónu s vstupmi *x* ich váhami *w*, k sume ktorých sa pripočíta tzv. bias, čo je vnútorná konštanta každého neurónu nakoniec sa tento výsledok predá aktivačnej funkcii, ktorej výsledkom je výstup z neurónu. Výstup môže byť vstupom pre ďalší neurón alebo môže predstavovať konečnú hodnotu, na základe ktorej bude prevedená klasifikácia. Učenie takejto siete prebieha správnym nastavením váh a biasu , ktoré by produkovali očakávaný výsledok.



Obrázok 3.1 Perceptrón (prepracované z [1])

3.2 Neurónová sieť backpropagation

Ako už bolo spomenuté neuróny sa spájajú do sietí a jednou z nich je aj sieť backpropagation v preklade sieť so spätným šírením chyby, ktorej znázornenie je zobrazené na obrázku 3.2. Táto sieť sa skladá z troch základných vrstiev a to vstupná vrstva, výstupná vrstva a skryté vrstvy, ktorých počet môže byť 0..N.



Obrázok 3.2 Model neurónovej siete backpropagation (prepracované z [1])

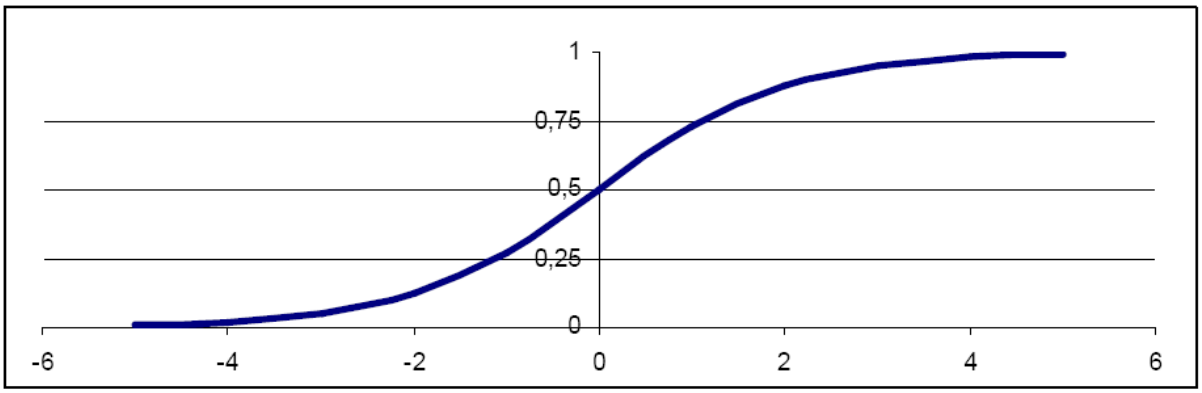
Na vstupnú vrstvu je privedený vzor dát, ktorý len šíri ďalej do skrytých vrstiev. Skrytých vrstiev môže byť viac, len jedna, prípadne žiadna. Poslednou je výstupná vrstva výstupné hodnoty z tejto vrstvy sú výsledkom pre vstupný vzor dát. Pri klasifikácii je to vhodne zakódovaná trieda, do ktorej má byť vzor klasifikovaný.

3.2.1 Princíp fungovania siete backpropagation

Na začiatku sú váhy a biasy jednotlivých neurónov nastavené na náhodné malé čísla. Na vstupnú vrstvu je privedený prvá vzorka dát. Vstupná vrstva tieto dáta len posunie ďalej do prvej skrytej vrstvy. Neuróny z tejto vrstvy vypočítajú sumu súčinov váh a vstupných hodnôt, ku ktorým sa pripočítajú jednotlivé biasy. Aktivačná funkcia neurónov v sieti backpropagation má spojitý charakter jej vzorec je

$$y = \frac{1}{1+e^{-x}} \tag{3.1}$$

a jej graf je uvedený na obrázku 3.3.



Obrázok 3.3 Aktivačná funkcia (prevzaté z [1])

Výsledok tejto funkcie je šírený do ďalších neurónov. Posledná výstupná vrstva vytvorí tzv. výstupný vektor, ktorý reprezentuje vzor dát. Ak je sieť v štádiu učenia dochádza k spätnému šíreniu chyby. Výsledný vektor je porovnaný s očakávaným vektorom pre danú vzorku dát. Spätne sa tak menia hodnoty váh a biasov až kým nedôjde k čo najväčšej podobnosti s očakávaným výstupným vektorom. Táto fáza učenia sa opakuje stále pre všetky vzorky vstupných dát, až kým sa sieť nenaučí na všetky z nich reagovať.

3.2.2 Algoritmus neurónovej siete backpropagation

V tejto podkapitole bližšie popíšem algoritmus neuónovej siete backpropagation. Viac o algoritme backpropagation, ako aj algoritmoch iných neurónových sietí je uvedené v [2]. Nižšie uvedený algoritmus 3.1 je taktiež prevzatý z [2].

Algoritmus 3.1 Algoritmus učenia neurónovej siete backpropagation

Vstup:

S = dáta z trénovacej množiny, neorónová sieť backpropagation

Výstup:

Naučená neurónová sieť backpropagation z množiny S

Metóda:

- Inicializuj všetky váhy a biasy v neurónovej sieti náhodnými malými hodnotami (napr. z intervalu <-1,1>)
- **while** (neurónov sieť nieje naučená) **do**
  - **for** (každý prvok *X* z trénovacej množiny) **do**
    - Postupne pre každý neurón *j* v jednotlivých vrstvách počítaj (vrstvy prechádzaj od prvej po poslednú):
      - $I_j = \sum_i w_{ij} O_i + \theta_j$, (3.2)

Kde O_i sú výstupy neurónou z minulej vrstvy, ktoré sú vstupom pre neurón s indexom j, w_{ij} je váha medzi neurónom i a aktuálnym neurónom j, θ_j je bias aktuálneho neurónu, I_j je vstup aktuálneho neurónu pre aktivačnú funkciu.

$$O_j = \frac{1}{1+e^{-I_j}},$$

kde I_j je vstup pre aktivačnú funkciu, O_j je výstup neurónu s indexom j.

// Spätné šírenie chyby:

- Pre každý neurón j výstupnej vrstvy vypočítaj:
  - $Err_j = O_j(1 - O_j)(T_j - O_j),$ (3.3)

  kde T_j je očakávaná hodnota na výstupe, O_j je skutočná hodnota na výstupe.
- Pre každý neurón j v skrytých vrstvách vypočítaj (vrstvy prechádzaj od poslednej k prvej):
  - $Err_j = O_j(1 - O_j)\sum_k Err_k w_{jk},$ (3.4)

  kde O_j je skutočná hodnota na výstupe, w_{jk} je váha medzi neurónom j a následným neurónom k.
- Všetky váhy oprav nasledovne:
  - $\Delta w_{ij} = (l) Err_j O_i$ (3.5)
  - $w_{ij} = w_{ij} + \Delta w_{ij}$ (3.6)
- Všetky biasy oprav nasledovne:
  - $\Delta\theta_i = (l) Err_j$ (3.7)
  - $\theta_i = \theta_j + \Delta\theta_j$ (3.8)

Poznámka: Vo vzorci na opravovanie váh a biasov sa vyskytuje výraz (l). Je to tzv. ***koeficient učenia***, ktorým je reálne číslo z intervalu <0,1>. Čím je toto číslo väčšie, tým ľahšie sa naučí neuonová sieť reagovať na aktuálnu vzorku dát, ale ľahšie „zabudne“ na ostatné. Odporúča sa tento koeficient dynamicky zmenšovať v závisosti na indexu aktuálnej iterácie.

4 Systém vyvíjaný na FIT

V tejto časti sa budem venovať systému pre dolovanie z dát vyvýjanému na Fakulte informačných technológií VUT v Brne. Tento systém je už niekoľko rokov vyvýjaný prostredníctvom bakalárskych a diplomových prác študentov. Dolovanie z dát zabezpečujú pripojiteľné moduly, ktoré dolujú a prezentujú dáta. Aplikácia pracuje na princípe klient-server, kde stranu servera reprezentuje databázový server Oracle s platformou Oracle Data Mining (ODM). Aplikácia je tvorená v jazyku Java v prostredí NetBeans, ktoré je použité ako modulárny základ aplikácie. V predchodzích rokoch sa na dolovanie z dát používal práve server Oracle, ktorý má v sebe zakomponované určité dolovacie metódy. V prípade neurónových sietí ODM neposkytuje podporu a tak dolovanie spolu sprípravou dát a ich transformáciou do portebnej formyzaisťuje klient. Server v tomto prípade slúži ako úložisko dát pre dolovanie. Pre ukladanie metadát a samotného procesu dolovania je použitý jazyk DMSL.

Aplikácia sa skladá z dvoch hlavných častí a tými sú jadro systému a moduly. Jadro je tvorené jadrom aplikácie, ktoré vytvoril Ing. Doležal a grafickú nadstavbu vytvoril Ing. Gálet. Tieto komponenty boli v ďaľších prácach upravené Ing. Krásným a Ing. Šebkem. Moduly sú postupne dopĺňané o ďalšie dolovacie metódy.

4.1 Použité technológie

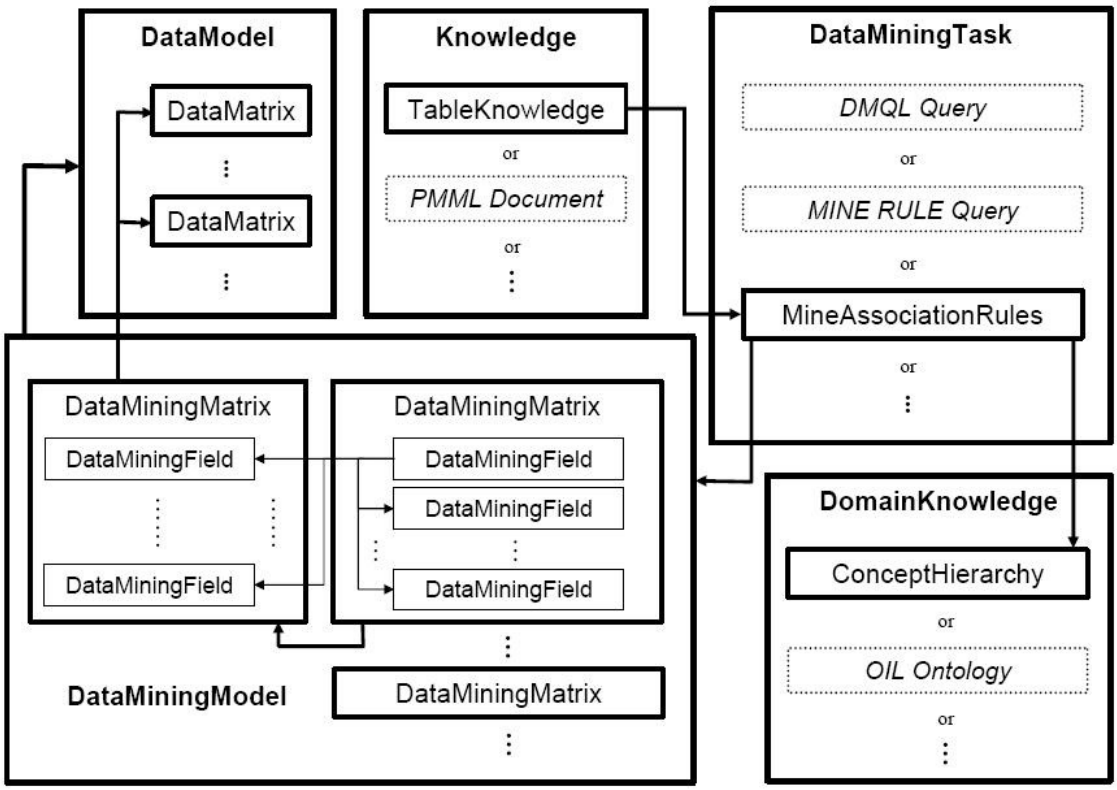
V tejto časti práce stručne opíšem technológie použité pri vytváraní dolovacieho modulu.

4.1.1 Jazyk DMSL

Jazyk DMSL bol vytvorený ako súčasť dizertačnej práce na FIT VUT v Brne Petrom Kotáskom [6]. Je to jazyk založený na štruktúre jazyka XML, zaoberajúci sa procesom získavania znalostí z databází. Definuje vstupné dáta, transformácie a rôzne iné komponenty dolovacieho procesu. V prípade mojej práce sa využíva hlavne na ukladanie metadát, ako sú parametre jednotlivých atribútov a parametre dolovacieho modulu. Pre potrebu aplikácie vyvíjanej na FIT VUT v Brne sú niektoré jeho časti upravené alebo úplne vynechané, keďže DMSL poskytuje komplexnejšiu podporu pre dolovanie z dát ako je portebná v prípade tejto aplikácie.

4.1.1.1 Štruktúra

Jazyk DMSL tvoria elementy, ktoré sú definované v DTD, podobne ako je tomu u ostatných jazykov vychádzajúcich z XML. Pre potrebu aplikácie je jazk DMSL vybavený elementom pre každú časť procesu získavanie znalostí z databáz (Obrázok 4.1).



Obrázok 4.1 Štruktúra jazyka DMSL (prevzaté z [6])

Teraz bližšie popíšem jednotlivé elementy:

- **DataModel** – tento element slúži k popisu vstupných dát pre daný dolovací proces. Tento element môže obsahovať ďalší element tzv. *dátových matíc (DataMatrix)*, ktorý reprezentuje výber dát v rámci jednej databázovej tabuľky. Každý stĺpec tabuľky je reprezentovaný elementom *DataField*, ktorý popisuje jeho vlastnosti. Môže sa odkazovať na element *FunctionPool.*

- **DataminingModel** – tento element popisuje tie dáta, ktoré udú použité pre samotný proces dolovania z dát, teda aj odvodené stĺpce a transformácie dát z *DataField. DataMiningModel* sa vždy odkazuje na práve jeden *DataModel,* ktorý pre neho reprezentuje zdroj vstupných dát. Ďalej sa tiež môže odkazovať na jeden, alebo žiadny *FunctionPool. DataMiningModel* obsahuje element *DataMiningMatrix,* ten sa skladá z *DataMiningFields.*

- **DomainKnowledge** – obsahuje informácie (znalosti) o dátach, ako sú integritné obmedzenia, ich vzťahy, obory hodnôt a ďalšie. Tento element má volnú syntax a v aplikácii vyvýjanej na FIT VUT v Brne sa zatiaľ nijak nevyužíva.

- **Knowledge –** tento element má taktiež voľnú syntax. Má dva povinné atribúty a tými sú použitý jazyk a meno. Slúži k reprezentácii získanej znalosti.

- **DataMiningTask -** tento element má taktiež voľnú syntax. Má dva povinné atribúty a tými sú použitý jazyk a meno.

- **FunctionPool** – atribút špecifikujúci funkcie, ktoré môžu byť aplikované na vstupné dáta pri ich čistení a predspracovaní. Každý *FunctionPool* má jednoznačné meno, pomocou ktorého sa na neho odkazujú ostatné elementy DMSL.

- **Header –** element , ktorý obsahuje nepovinné informácie o projekte (napr. autor, čas vytvorenia).

Z týchto elementou je poskladaný hlavnýelement DMSL, ktorý je v tvare:

```
< ! ELEMENT DMSL (HEADER?, (FUNCTIONPOOL | DATAMODEL | DATAMININGMODEL
|DOMAINKNOWLEDGE | DATAMININGTASK | KNOWLEDGE) +) >
```

Zo zápisu je zjavné že element *Header* ako jediný nemusí byť súčasťou záznamu DMSL. V tejto časti som podal len základné informácie o jazyku DMSL a jeho použitia. Podrobnejšie informácie sú uvedené v práci pána Kotáska [6].

4.1.2 NetBeans

Platforma NetBeans je open-source projekt zaisťovaný a sponzorovaný spoločnosťou Sun Microsystems. V súčasnosti je vo verzii 6.8. Projekt je rozdelený do dvoch hlavných častí:

1. Vývojové prostredie NetBeans (NetBeans IDE).
2. Vývojová platforma NetBeans (NetBeans Platform).

Netbeans IDE je napísané v jazyku Java a je aj postavené na platforme NetBeans. Je určené na vývoj aplikácií v jazyku Java, ale podporuje aj iné programovacie jazyky. V Jave podporuje všetky 3 jej hlavné platformy – J2SE, J2EE, J2ME. Obe časti NetBeans sú bezplatne šíriteľný produkt, ktorý je možné používať na operačných systémoch Windows, Linux, Mac OS X a Solária. Vývojové prostredie NetBeans IDE 6.5 obsahuje už aj vývojovú platformu NetBeans takže pre vývoj aplikácie a zaistenie jej modulárnosti nieje potrebné používať NetBeans Platform.

4.2 Popis systému

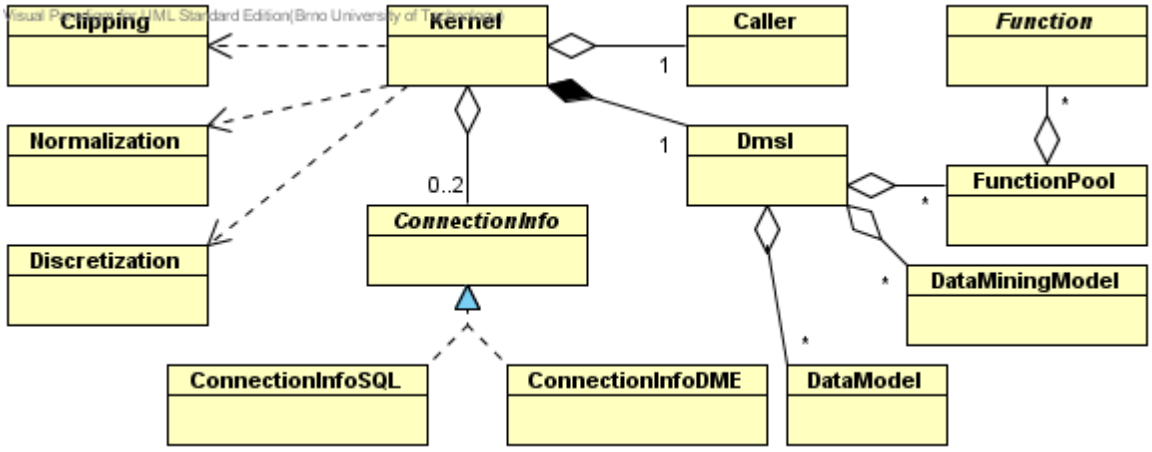
V tejto kapitole sa zamerám na popis hlavných častí systému. Konkrétnejšie informácie je možno získať z prác Ing. Krásného [8] a Ing. Šebka [9], alebo z prác , ktoré sa venujú jednotlivým modulom.

4.2.1 Jadro systému

Základnou triedou jadra je trieda *Kernel.java* (Obrázok 4.2). Táto trieda umožňuje pristupovať k triedam pre prácu s DMSL súborom (trieda *Dmsl.java*), prístup k zkompilovaným funkciám (trieda *Caller.java*) a pripojenie k serveru (trieda *ConnectionInfo.java*). Inštancie tried Clipping, Normalization a Discretization sa vytvárajú pri volaní metódy obsluhujúcej spustenie komponenty Transformations. Zaisťujú inicializáciu parametrov pre server a ich odosielanie.

Jadro teda poskytuje tieto funkcie a umožňuje:

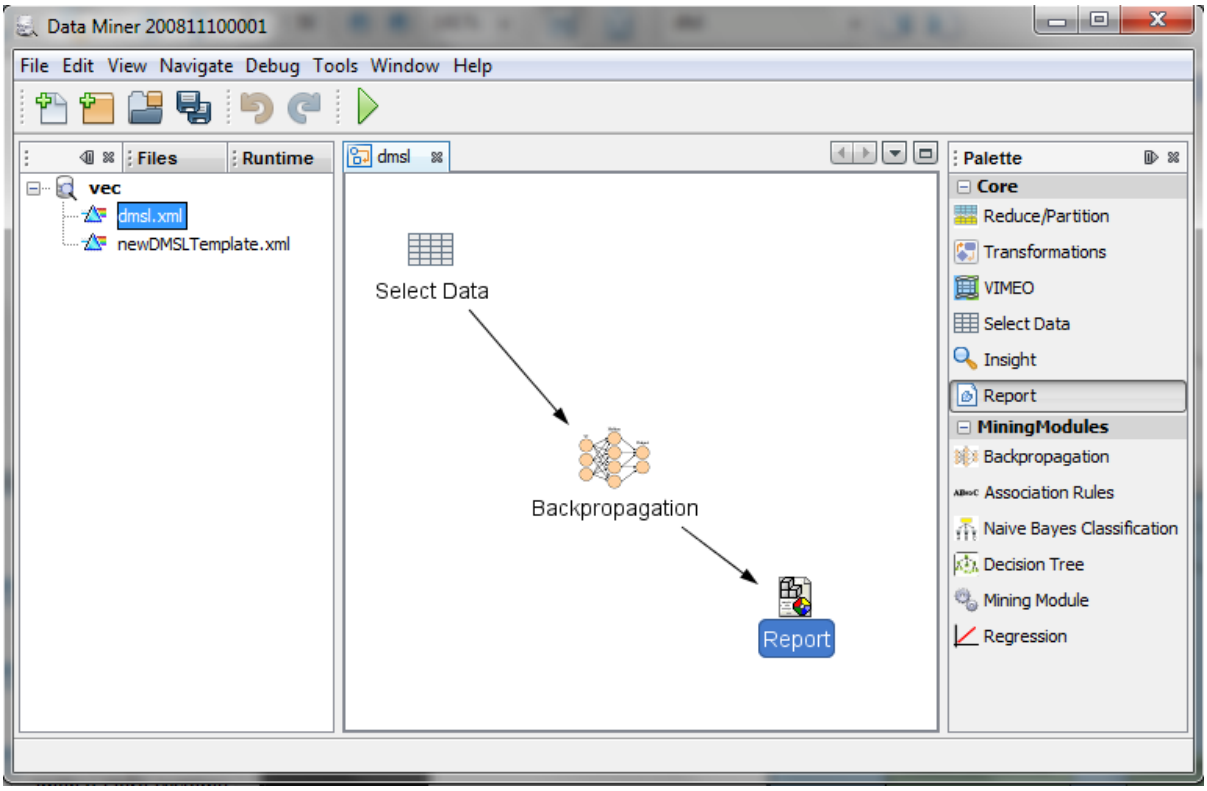
- Zadávať dolovacie metódy,
- načítanie a ukladanie metadát do DMSL,
- organizovanie práce v projekte,
- prevádzať transformácie dát,
- pridávať dolovacie moduly.



Obrázok 4.2 Diagram základných tried jadra(prevzaté z [9])

4.2.2 Grafické užívateľské rozhranie

Grafické užívateľské rozhranie je rozdelené do niekoľkých častí (Obázok. 4.3). V ľavej časti sa nachádza prieskumník projektov, kde je hrubo vznačený ten projekt, ktorý je práve aktuálny. Pod ním je pohľad na dolovaciu úlohu. Prostredná časť (pracovná plocha) predstavuje priestor pre vytváranie dolovacej úlohy. V pravej časti je umiestnená paleta s jednotlivými komponentami dolovacieho procesu. Komponenty možno jednoducho pretiahnutím umiestnit na pracovnú plochu a prepojením šípkami z jednej komponenty do druhej vznikne graf dolovacej úlohy, ktorý sa zaznamenáva do DMSL.



Obrázok 4.3 GIU aplikácie Data Miner

Komponenty:

- **Select Data –** komponenta pre výber vstupných dát. Obsahuje záložky pre výber zdroja dát. V záložke Columns Selection môže užívateľ ľubovoľne vyberať stĺpce, v ktorých sú dáta, ktoré majú byť použité v dolovacej úlohe. Je tu tiež záložka pre vkladanie podmienok pre spojovanie tabuliek, záložka pre nastavenie referencií a záložka pre import dát zo súboru typu CSV.
- **Transformations** – poskytuje možnosť aplikácie ODM na transformáciu, normalizáciu a diskretizatizáciu na vstupné dáta. Tiež umožňuje nezahrnúť do výstupu vstupné stĺpce a pomocou funkcií odvodiť stĺpce nové. Zaisťuje prístup k editáciám funkcií.
- **Vimeo –** táto komponenta funguje ako dátový filter. Všetky výstupné stĺpce sú rovnaké ako vstupné, ale je k nim piradená nejaka *vimeo* funkcia. Neumožňuje stĺpce premenovávať. Zaisťuje prístup keditáciám funkcií.
- **Reduce/Partition** – táto komponenta rozdeľuje dáta pre klasifikačné a prediktívne dolovacie moduly. Dokáže znížiť počet záznamov ako aj rozdeliť dáta na trénovacie a testovacie.
- **Report –** komponenta, ktorá slúži na zobrazenie výsledkov dolovacieho procesu v mojom prípade ide o zobrazenie grafu chyby učenia ako aj záložku na import nových dát, ktoré majú byť klasifikované natrénovanou neurónovou sieťou backpropagation. Taktiež dokáže exportovať dáta po klasifikácii do súboru typu *CSV*.

- **Backpropagation –** komponenta pre dolovací klasifikačný modul založený na neurónovej sieti backpropagation.

Komponenty Backpropagation a Report boli predmetom mojej bakalárskej práca budem sa im venovať v kapitole 5.2.1.

5 Implementácia modulu

Pre impelemnáciu modulu založeného na neurónovej sieti backpropagation je nutné vedieť, ako sa daný modul správa, ako sa zapíše do DMSL a ako ho správne začleniť do aplikácie. V nasledujúcich kapitolách sa pokúsim vysvetliť jeho implementáciu, začleneni do aplikácie ako aj prezentáciu vysledku v podobe komponenty Report.

5.1 Začlenenie modulu do aplikácie

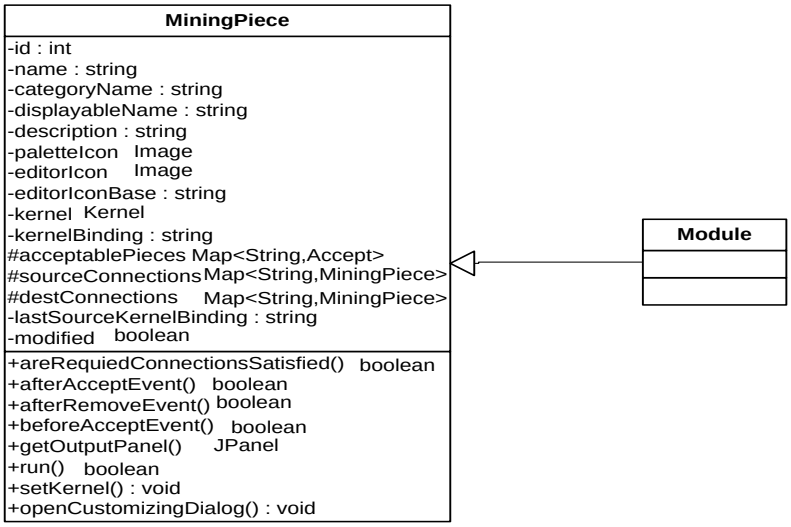
Spôsob vytvorenia modulu je popísaný v [8]. V tejto kapitole preto popíšem len základný princíp nevyhnutný pre pochopenie začlenenia modulu do aplikácie. Pre vývoj modulu je potrebné vo vývojovom prostredí NetBeans vytvoriť nový projekt *Module* z *NetBeans Modules*. V prípade modulu backpropagation mám k dispozícii zdrojové kódy celej aplikácie tak môžem vybrať možnosť Add to Module Suite. Keby som nemal k dispozícii zdrojové kódy aplikácie bolo by potrebné vybrať druhú možnosť *Standalone Module*. Ďalej len vyplníme *Code Name Base* a názov nového modulu a vytvorenie nového modulu je dokončené.

Následne je potrebné v zložke nového modulu vybrať zložku *Libraries* a v nej pridať závislosti novéhomodul pomocou volby *Add Module Dependency*. Z ponuky je potrebné vybrať možnosti: *Dialogs API, dm-api, dm-core-wrapper, dm-jfreechart-wrapper, dm-jmath-wrapper, UI Utilities API* a *Utilities API*. Všetky tieto závislosti sú potrebné pre ďaľší vývoj nového modulu.

Teraz je už možné začleniť nový modul do aplikácie dataminer a postupne začať implementovať jeho funkčnosť. Predtým jeale nutnosť implementovať abstraktnú triedu *MiningPiece* a upraviť záznam v súbore *layer.xml*. Tejto problematike sa budem venovať v nasledujúcich kapitolách.

5.1.1 Implementácia abstraktnej triedy MiningPiece

Pre všetky moduly je potrebné implementovať abstraktnú triedu *MiningPiece* (Obrázok 5.1), pretože iba trieda, ktorá dedí vlastnosti tejto triedy, sa dá vložiť do aplikcie ako nová komponenta.



Obrázok 5.1 Trieda MiningPiece

Pre správnu funkčnosť nového modulu je potreba implementovať všetky metóty tejto abstraktnej triedy. Bližšie popíšem tie, ktoré sa priamo podieľajú na dolovacom procese.

- **opencustomizingDialog() –** vytvára a otvára panel, do ktorého sa zadávajú parametre modlu. Táto metóda je volaná pri otvorení panelu *Backpropagation.*
- **getOutputPanel() –** metóda vytvára panel štandardnej komponenty Report, v ktorej sa zobrazujú výsledy dolovacieho procesu.
- **run() –** spúšťa akciu komponenty. V prípade dolovacieho modulu spúšťa samotný dolovacý proces.

V konštruktore triedy, ktorá dedí od *MiningPiece* je nutné nastaviť nasledujúce:

- setDisplayableName(NbBundle.getMessage(Backpropagation.class, "NAME_Backpropagation"));
- setDescription(NbBundle.getMessage(Backpropagation.class, "HINT_Backpropagation"));
- setPaletteIcon("cz/vutbr/fit/dataminer/NeuralNetwork/resources/module16.png");
- setEditorIcon("cz/vutbr/fit/dataminer/NeuralNetwork/resources/module48.png");

Prvý riadok hovorí , že meno modulu sa nacháda v lokalizačnom balíku "*NAME_Backpropagation*". Rovnako druhý riadok hovorí že nápoveda je uložená v lokalizačnom balíku "*HINT_Backpropagation*". Posledné dva body udávajú cestu s ikonám modulu. V treťom bode je to cesta k ikone, ktorá bude zobrazená na palete nástrojov v pravej časti aplikácie a je uložená v súbore *module16.png.* Štvrtý bod udáva cestuk ikone, ktorá bude zobrazená v grafe dolovacieho procesu a jej názov je *module48.png.* Konštruktor triedy teda iba nastaví názov, nápovedu a cesty k ikonám dolovacieho modulu.

Aby aplikácia Dataminer našla popis mena a nápovedy je treba v súbore *Bundle.properties* tieto názvy nastaviť. Súbor *Bundle.properties* preto môže obsahovať napríklad záznam:

- NAME_Backpropagation=Backpropagation
- HINT_Backpropagation=Mining module for classification by backpropagation

5.1.2 Integrácia vytvoreného modulu do aplikácie

Teraz keď už sú všetky potrebné metódy implementované môžeme modul integrovať do aplikácie. To sa prevedie pomocou vytvorenie záznamu v súbore *layer.xml*. Tento súbor predstavuje virtuálny systém súborov, ktorý sa z rôznych modulov integruje do jedného celku. To znamená, že záznamy, ktoré tento modul vytvorí, budú viditeľné aj z ostatných modulov. Jednotlivé moduly o sebe navzájom nič nevedia ani nemajú prístup k svojim triedam, aj keď bežia v rámci jedného virtuálneho stroja Javy. O vytváranie inštancií sa stará systém *FileSystem*, ktorý poskytuje inštancie ostatným modulom [8]. Pre integráciu modulu do systému je nutné do súboru *layer.xml* pridať nasledujúci záznam:

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

Táto časť kódu vytvorí inštanciu triedy modulu a zaregistruje ho v *MiningPieceRegistry* do kategórie *Mining Modules,* čoho následok bude, že komponenta bude zobrazovaná v palete modulov v sekcii Mining Modules.

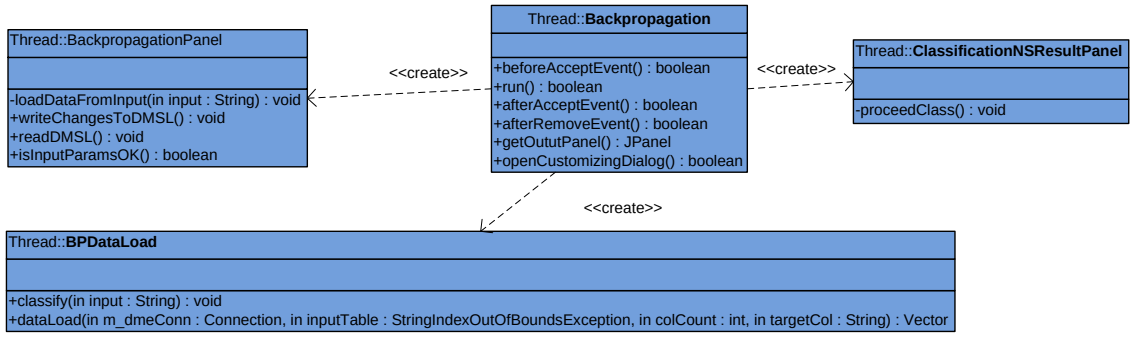
Ďalšia časť určuje množinu komponentov, na ktoré je možné modul pripojiť a ďalej, ktoré komponenty je možné na modul pripojiť. V prípade modulu backpropagation je možné ho pripojiť na komponenty *Select, TransF, VimeoF* a *Reduce.* Jediným komponentom, ktorý je možné pripojiť na modul backpropagation je komponenta *Report*, ktorá zobazuje výsledky dolovacieho procesu.
Zvyšná časť záznamu v *Filesystem*:

```
 <folder name="MiningPieceConfig">
  <folder name="cz-vutbr-fit-dataminer-NeuralNetwork-Backpropagation">
   <folder name="accept">
    <file name="cz-vutbr-fit-dataminer-editor-palette-items-Select"/>
    <file name="cz-vutbr-fit-dataminer-editor-palette-items-Transf"/>
    <file name="cz-vutbr-fit-dataminer-editor-palette-items-VimeoF"/>
```

```
        <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-NeuralNetwork-Backpropagation"/>
      </folder>
    </folder>
  </folder>
</filesystem>
```

5.2 Modul backpropagation

V tejto kapitole popíšem iplementáciu dolovacieho modulu na princípe nerónovej siete backpropagation. Na obrázku 5.2 je zobrazené znázornenie hlavných tried modulu backpropagation, z ktorých má špeciálnu pozíciu trieda Backpropagation, ktorá rozširuje triedu *MiningPiece*, ktorá ako už bolo povedané zaisťuje pripojenie modulu k aplikácii Dataminer. Implementáciou tejto triedy sa zaoberá ďalšia kapitola. Mimo metód dedených z triedy *MiningPiece* zabezpečuje trieda *Backpropagation* aj prečítanie dát z DMSL pri otvorení už existujúceho dolovacieho procesu.



Obrázok 5.2 Znázornenie hlavných tried modulu backpropagation

5.2.1 Implementácia triedy rozširujúcej triedu MiningPiece

V tejto kapitole podrobnejšie popíšem implementáciu metód triedy *MiningPiece*, ktoré od nej dedí trieda *Backpropagation.*

- **beforeAcceptEvent()** – v tejto metóde sa len vygeneruje id komponenty aby bolo možné komponentu odlíšiť od ostatných komponent grafu dolovacieho procesu, ako aj do grafu dolovacieho procesu vkladať viac komponent typu *Backpropagation.*

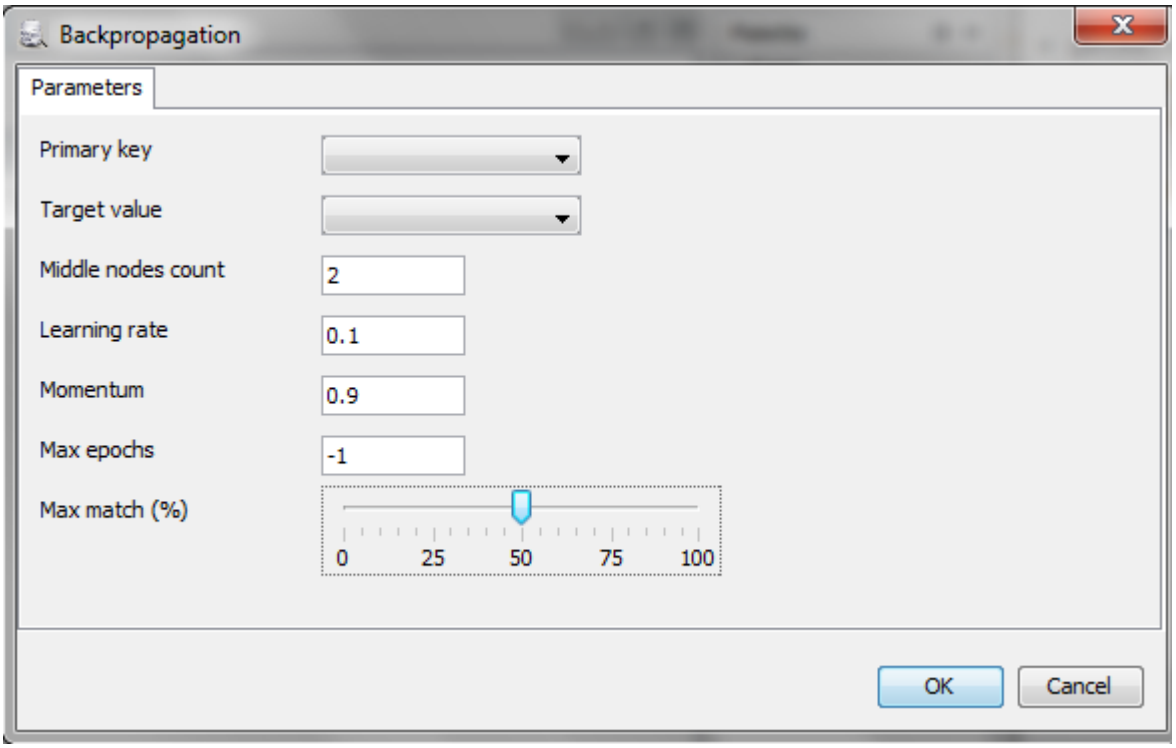
- **afterAcceptEvent() –** metóda sa volá po vložení komponenty do grafu dolovacieho procesu. Po vložení komponenty *Backpropagation* do grafu dolovacieho procesu je potrebné vytvoriť záznam v DMSL. V DMSL sa vytvorí nasledujúci záznam:

```
<DataMiningTask name=\"Backpropagation"\" type=\"Backpropagation\"
language=\"XML\">
<Backpropagation algorithm=\"Backpropagation\" >
        <InputInfo primaryKey=\" null\" target=\"null\" />
        <Parameters>
                <Middlenodescount>2</Middlenodescount>
                <Learningrate>0.1</Learningrate>
                <Momentum>0.9</Momentum>
                <MaxEpochs>-1</MaxEpochs>
                <MaxMatch>100</MaxMatch>
        </Parameters>
        <ChartPoints>
        </ChartPoints>
</Backpropagation>
</DataMiningTask>
```

Týmto sa vytvorí základ elementu *DataMiningTask*, k neskoršiemu uloženiu potrebných parametrov. Parametre sú nastavené na počiatočné hodnoty, ktoré sa zobrazia pri prvom otvorení panelu Backpropagation. Táto komponenta teda len vytvorí záznam v DMSL, ktorý sa bude v priebehu dolovania meniť.

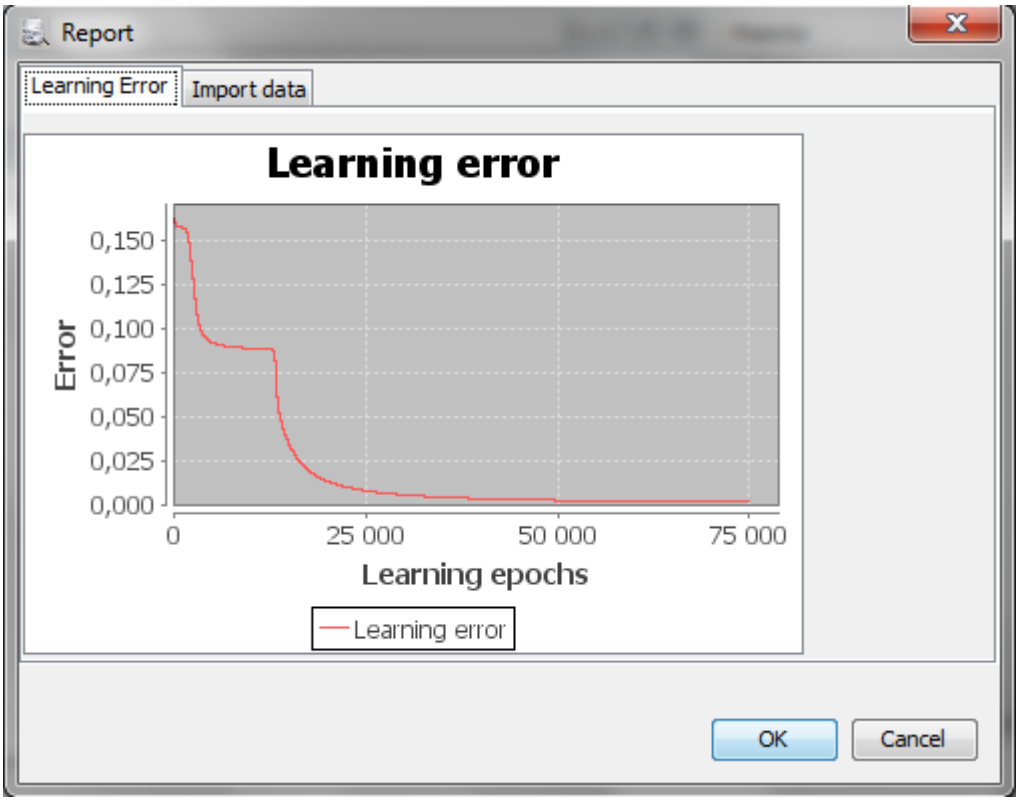
- **openCustomizingDialog –** táto metóda vytvára a zobrazuje panel, do ktorého užívateľ zadáva parametre dôležité pre dolovací proces. Panel obsahuje sedem parametrov, ktoré je potrebné zadať. Panel je zobrazený na obrázku č.5.3.
    - **Primary key** - udáva primáry kľúč tabuľky z predošlej komponenty grafu. Tento parameter nieje pre samotný dolovací proces nevyhnutný, keďže tabuľka pre dolovanie nemsí primárny kľúč mať, avšak prispieva k rýchlejšiemu behu programu, keďže indexácia prvkov je robená cez tento parameter.
    - **Target Value** - druhý parameter je už pre dolovanie nevyhnutný keďže udáva cieľový atribút, teda atribút, ktorý udáva triedy, do ktorých sa bude klasifikovať. Opäť je možné vyberať z názvou atribútou, ktoré boli vybrané v predošlej komponente grafu dolovacieho procesu.

- **Middle nodes count –** tento parameter udáva počet neurónov v skrytej vrstve, užívateľ zadá celé číslo, podľa uváženia. Čím viac je neurónov v skrytej vrstve tým viac sa vykoná výpočtov v jednom prechode neurónovou sieťou.
- **Learning rate** – tento parameter udáva krok, s ktorým sa menia váhy a biasy pri prenose chyby. Typicky sa uáva hodnota z intervalu <0,1>, najčastejšie hodnota v intervale <0,05;0,75>.
- **Momentum –** tento parameter súvisí s parametrom learning rate. Keď je learning rate príliš malé je učenie siete príliš pomalé a naopak ak je learning rate príliš veľké učenie siete je nepresné. Týmto vznikajú oscilácie siete. Tlmeniu tejto oscilácie pomáha práve konštanta momentum, ktorá sa pridá do výpočtu zmeny váh a jej zavedením sa teoreticky zvýši rýchlosť učenia. Jej hodnota sa uáva z intervalu <0,1>, pričom udáva sa hodnota blízko nule, alebo blízko jednej.
- **Max epochs –** ako už názov napovedá ide o parameter, ktorý udáva maximálny počet prechodov celou sieťou. Udáva sa celým kladným číslom a v špeciálnom prípade, ak má byť počet prechodov sieťou neobmedzený, udáva sa hodnota -1.
- **Max match –** hodnota tohto parametru udáva tiež ukončujúcu podmienku učenia. Jedná sa o percentuálne vyjadrenie správne klasifikovaných tried. Teda ak bude max match 100 % sieť sa bude učiť pokiaľ neklasifikuje všetky vzory správne.



Obrázok 5.3 Zobrazenie panelu parametrov

- **run()** – táto metóda spúšťa samotný dolovací proces. Najprv ale prečíta parametre z DMSL. Tieto parametre nečíta z panelu ale priamo z DMSL z jediného dôvodu, že panel nemusel byť pre spustenie dolovacieho procesu vôbec otvorený a preto panel po stlačení tlačídla *OK* len zapíše parametre do DMSL a metóda *run* si ich pre spustenie dolovacieho procesu len prečíta. Metóda vytvorí inštanciu objektu *BPDataLoad*, ktorý načíta dáta a vytvorí neurónovú sieť podľa vzoru dát.
- **getOutputPanel()** – táto metóda prečíta z DMSL uložené dáta o dolovacom procese a vyvolá spustenie panelu komponenty *Report*. Panel obsahuje 2 záložky v prvej je zbrazený vývoj chyby učenia v závislosti od počtu prechodov cez neurónovú sieť. Druhá záložka slúži na import tabuľky, ktorú chceme klasifikovať pomocou už natrénovanej siete. Táto záložka tiež dokáže exportovať výsledky do súboru CSV. Príklad tejto komponenty ukazuje obrázok 5.4



Obrázok 5.4 Komponenta Report

- **afterRemoveEvent** – táto metóda sa volá po odstránení komponenty z grafu dolovacieho procesu. V tomto prípade len odstráni z DMSL elementy *DataminingTask* a *Knowledge*.

5.2.2 Triedy dôležité pre dolovací proces

V tejto kapitole popíšem ostatné triedy dôležité pre proces dolovania pomocou modulu backpropagation.

5.2.2.1 BPDataLoad

Táto trieda načíta dáta z databázy do vektoru tried *InpuClumn*. Trieda *InputColumn* zaznamená všetky dôležité dáta o jednom stĺpci tabuľky ako sú hodnoty jednotlivých položiek, ich počet ako aj počet rôznych hodnôt keďže tieto sú nevyhnutné pre správne vytvorenie neurónovej siete. Trieda *BPDataLoad* po tom, ako načíta dáta vyvorí z nich inštanciu triedy *NeuralSite*, ktorá vyvorí samotnú neurónovú sieť podľa vzoru dát.

5.2.2.2 NeuralSite

Táto trieda ako už bolo spomenuté vytvorí samotnú neurónovú sieť. Avšak to nie je jej jedinou úlohou, tiež zaznamenáva zmenu chyby učenia neurónovej siete pre panel s vysledkami dolovacieho procesu.

5.2.3 Použité externé knižnice

V tejto kapitole popíšem externé knižnice, ktoré som využíval pri implementácii.

5.2.3.1 Backprop

Jedná sa o sieť backpropagation, ktorá bola použitá pri implementácii. Je vytvorená Guyom Colom v rámci projektu Digital Burro [10]. Je pod licenciou Pubic domain, čo znamená, že pán Cole pre ňu neposkytuje žiadnu podporu ani garanciu. Jej výsledky sú avšak vyskúšané a bezchybné. Túto sieť som mierne upravil pre potreby implementácie. Hlavne z dôvodu zaznamenávanie chyby učenia. Sieť podporuje aj spojité hodnoty atribútov avšak pre naše potreby sú tieto hodnoty už diskretizované.

5.2.3.2 JfreeChart

Jedná sa o knižnicu používanú v rámci projektu Dataminer na vykresľovanie dvojdimenzionálnych grafov. Je taktiež úplne zadarmo a je šírená pod licenciou GNU Lesser General Public Licence (LGPL). Podporuje širokú škálu vstupov a výstupov a je podrobnejšie opísaná v [11].

5.2.4 Problémy spojené s implementáciou

V tejto kapitole sa budem venovať niektorým problémom, ktoré sa vyskytli v spojení s implementáciou.

5.2.4.1 Vyrvorenie neurónovej siete na vzorke dát

Keďže používam už implementovanú neurónovú sieť musel som dáta transformovať do podoby aké si implementácia siete vyžadovala. Z každej hodnoty atribútu bolo treba vytvoriť pole hodnôt, ktoré predstavovali jednotlivé neuróny. Tieto polia som musel zaznamenávať, keďže hodnoty atribútov sa mohli v jednom stĺpci opakovať. Toto som vyriešil v triede *InputColumn*, kde okrem vytvárania polí vytváram aj vektor hodnôt, ktoré sa v danom stĺpci vyskytli. Musel som zaznanemávať aj počet rôznych hodnôt, ktoré udávali počet neurónov pre daný atribút. Toto malo jednu výnimku a to ak boli v stĺpci len dve rôzne hodnoty atribútu, pre tieto sa vytvoril len jeden neurón, ktorý hovoril či je to jedna alebo druhá hodnota. Z môjho pohľadu bol toto jediný relevantný problém, ktorý sa pri implementácii vyskytol.

6 Príklad možného použitia modulu

Vtejto kapitole ukážem možné použitie modulu krok za krokom.

6.1 Vytvorenie projektu v Datamineri

Po spustení aplikácie Dataminer je nutné vytoriť nový projekt. To vykonáe pomocou menu File-> New Project, prípadne pomocou klávesovej skratky Ctrl + Shift +N. Následne pomocou sprievodcu vyberieme projekt *Data Mining Project* z kategórie *Knowledge Discovery* a klikneme na tlačítko Next. Ďalej potom pomenujeme nový projekt a vberieme umiestnenie jeho uloženia a znovu stlačíme tlačidlo Next. Na záver vybereme *New Oracle database connection* a nastavíme údaje slúžiace k pripojeniu k serveru.

Login : dmuser6

Password : besek

Connection url : jdbc:oracle:thin:@berta.fit.vutbr.cz:1521:STUD

Na záver to len potvrdíme tlačítkom Finish. Tým sme vytvorili prázdny projekt v aplikácii Dataminer.

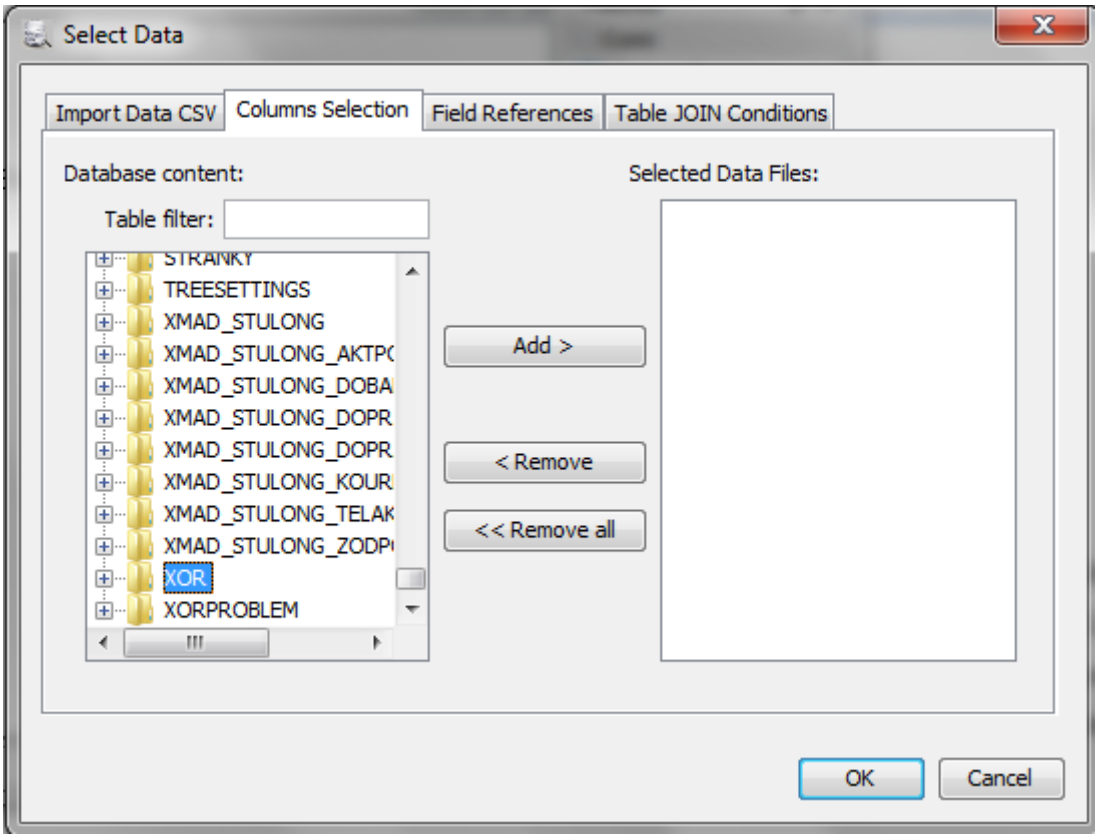
6.2 Vytvorenie grafu dolovacieho procesu

Keď už máme vytvorený prázdny projekt je treba vytoriť dolovaciu úlohu. To sa vykoná pomocou vytvorenia grafu dolovacieho procesu. Komponenty, ktoré sú nevyhnutné pre vytvorenie dolovacej úlohy sú *Select Data, Backpropagation* a *Report.* Tie je nutné prepojiť ako na obrázku 6.1.

Obrázok 6.1 Ukážka dolovacieho procesu

6.3 Výber dát

Po otvorení komponenty na výber dát (Obrázok 6.2) máme viac možností ako tieto dáta vybrať. Môžme ich importovať zo súboru CSV, alebo vybrať z tabuliek už vytvorených v databáze. V záložke Columns selection vyberieme napr. tabuľku XOR, ktorá obsahuje dáta o funkcii XOR. Túto funkciu spomínam zámerne, keďže sa jedná o nelineárny problém a dá sa na nej dobre dokázať že neurónové siete vedia riešiť aj nelineárne problémy. Ďalej vyberieme stĺpce, ktoré chceme pre dolovaciu úlohu použiť, v tomto prípade sú to všetky stĺpce tabuľky.



Obrázok 6.2 Ukážka komponenty Select Data

Táto komponenta umožňuje aj iné funcie, ktoré sú spomenuté v kapitole 4.2.2.

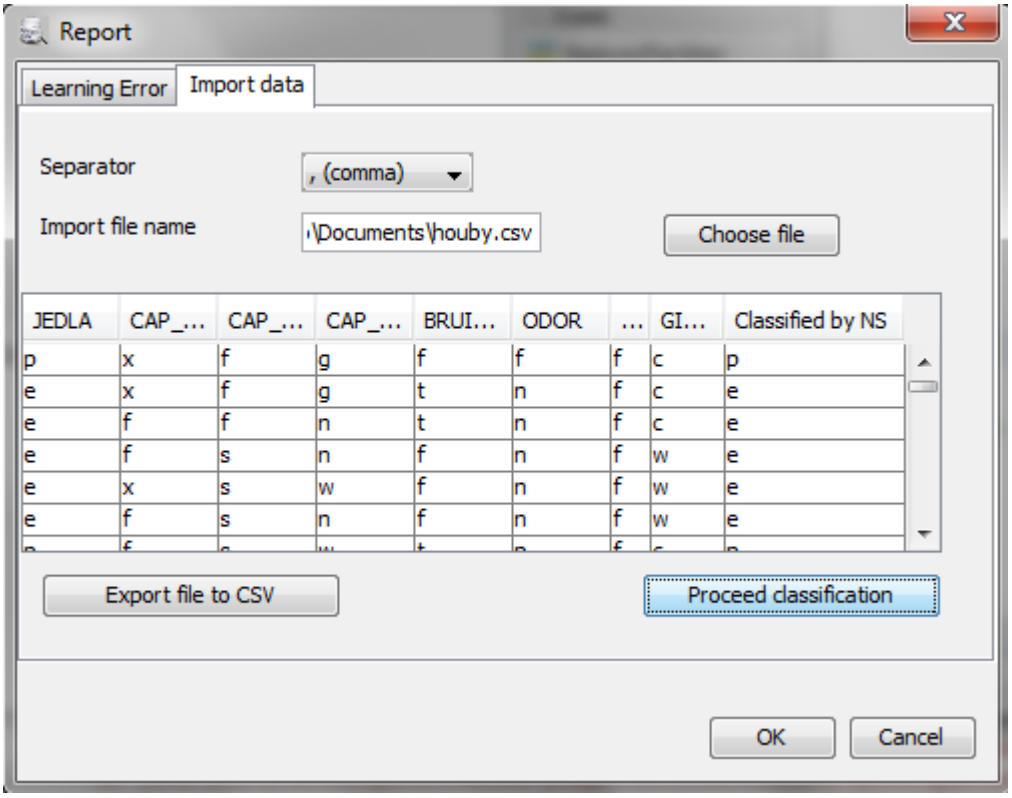
6.4 Nastavenie parametrov pre dolovanie

Komponenta bola bližšie popísaná v časti 5.2.1 pri volaní funkcie openCustomizingDialog. Pre prehľad uvediem len stručný popis parametrov v komponente.

- **Primary key** – primárny kľuč tabuľky (nepovinné).
- **Target value** – cieľový atribút.
- **Middle nodes count** – počet uzlov skrytej vrstvy.
- **Learning rate** – konštanta, pridávaná pre výpočet váh.
- **Momentum** – konštanta urýchľujúca výpočet váh.
- **Max epochs** – maximálny počet cyklov neurónovej siete.
- **Max match** – presnosť klasifikácie z pohľadu percentuálneho vyjadrenia úspešnosti klasifikácie.

6.5 Komponenta report

Táto komponenta zobrazuje výsledky dolovania pomocou grafu krivky chyby učenia. Avšak dokáže aj načítať dáta a aplikovať na ne klasifikáciu neurónovou sieťou ako ukazuje obrázok 6.3. Posledný stĺpec je výsledok klasifikácie neurónovou sieťou pri cieľovom atribúte JEDLA.



Obrázok 6.3 Ukážka komponenty report

7 Zhodnotenie výsledkov

Neurónové siete sú závislé od správnosti zadaných parametrov a dát. Ak napríklad zadám problém, ktorý sieť nedokáže vyriešiť bude sa snažiť donekonečna učiť avšak nikdy nebude dávať správne výsledky. Preto je potrebné vstupné dáta správne zadať ako aj správne nastaviť parametre. Ako príklad uvediem už spomínaný problém XOR ak zadám ako cieľový atribút stĺpec X sieť nebude schopná sa naučiť relevantne reagovať, keďže ako ukazuje obrázok 7.1 z hodnôt Y a output sa nedá odvodiť hodnota X.

X	y	output
1	1	0
1	0	1
0	1	1
0	0	0

Obrázok 7.1 tabulka XOR

Neurónové siete všeobecne vykazujú dobré výsledky pri správne zadanom modele siete či už ide o počet neurónov skrytej vrstvy alebo parametre learning rate a momentum. Toto sa dá dosiahnuť len testovaním siete a zmenou týcho parametrov. Keďže rýchlosť je u algoritmov dolovania z dát významný faktor v nasledujúcich testoch sa budem snažiť zamerať na vplyvy zmien počtu atribútov, počtu neuónov v skrytej vrstve ako aj vplyv learning rate na časovú náročnosť implementovaného algoritmu. Na toto mi poslúži tabuľka Houby_Lepiota_ZZN, v ktorej sú záznamy o hubách. Túto tabuľku som vybral z dôvodu, že na ňu dokáže neurónová sieť dobre reagovať už pri malom počte atribútov a záznamov a tak je veľký priestor pre experimentovanie s ňou. Je tu 24 atribútov z toho ako cieľový si vyberiem atribut *JEDLA*, ktorý udáva či je huba jedlá alebo nie. Ostatné atribúty budeme klasifikovať teda do dvoch tried jedlá, alebo nejedlá. Najdôležitejší atribút v klasifikácii je atribút *ODOR*, ktorý, ako sa z dolovania zistilo má jediný vplyv nato či je huba jedlá alebo nie. Ostatné atribúty poslúžia na vytváranie zložitejšej neurónovej siete podľa potreby testov. Treba však uviesť že, neurónová sieť závisí čiastočne aj od počiatočného náhodného nastavenia váh a biasov, ktoré môže ovplyvniť čas klasifikácie.

Číslo pokusu	Neuróny skrytej vrstvy	Počet záznamov	Počet atribútov	Počet cyklov	Čas klasifikácie (s)
1.	3	4097	23	61	11,7
2.	3	4097	23	120	16,3
3.	3	4097	23	86	12,6
4.	3	4097	23	92	13,05
5.	3	4097	23	136	16,42
6.	10	4097	23	99	30,6
7.	10	4097	23	186	52,21
8.	10	4097	23	140	41,21
9.	10	4097	23	98	30,58
10.	10	4097	23	165	45,92
11.	3	2050	23	89	6,45
12.	3	2050	23	166	8,98
13.	3	2050	23	115	7,68
14.	3	2050	23	93	6,53
15.	3	2050	23	154	8,89
16.	10	2050	23	249	30,56
17.	10	2050	23	69	11,06
18.	10	2050	23	168	21,82
19.	10	2050	23	228	29,82
20.	10	2050	23	154	20,59
21.	3	4097	11	72	4,33
22.	3	4097	11	66	8,75
23.	3	4097	11	87	9,23
24.	3	4097	11	62	8,70
25.	3	4097	11	96	9,52
26.	10	4097	11	63	12,93
27.	10	4097	11	57	12,62
28.	10	4097	11	43	12,34
29.	10	4097	11	52	12,58
30.	10	4097	11	76	13,03
31.	3	2050	11	48	4,23
32.	3	2050	11	31	4,05
33.	3	2050	11	91	4,8
34.	3	2050	11	51	4,24
35.	3	2050	11	78	4,58
36.	10	2050	11	95	8,04
37.	10	2050	11	105	8,61
38.	10	2050	11	92	8,19
39.	10	2050	11	147	10,97
40.	10	2050	11	78	7,3

Tabuľka 7.1 Testy klasifikácie

Tabuľka 7.1 ukazuje výťah z prevedených testov a to konkrétne 40 testov pre rôzne nastavenia vstupných parametrov, ktorými sú počet neurónov skrytej vrstvy, počet záznamov, počet atribútov. Ako je vidno na prvý pohľad každý z týchto parametrov má vplyv na dĺžku výpočtu ako aj

na počet cyklov. Všeobecne platí pravidlo, že s narastajúcim počtom každého z parametrov narastá aj počet výpočtov a teda sa zvyšuje aj doba výpočtu. Keďže pokusov bolo len 40 výsledy sú skreslené vplyvom už spomínaných nastavení váh a biasov. Tabuľka 7.2, tabuľka 7.3 a tabuľka 7.4 ukazujú priemerné hodnoty z testovania.

Zhodnotenie vplyvu parametrov na testovací proces:

- **Vplyv počtu neurónov skrytej vrstvy –**

Počet neurónov	Priemer počtu cyklov	Priemerný čas
3	89,7	8,55
10	118,2	21,05

Tabuľka 7.2 Vplyv počtu neurónov

S narastajúcim počtom neurónou narastá aj počet výpočtov a tým aj čas. Zaujímavý údaj je počet cyklov, ktorý je potrebný pre naučenie siete. Vo väčšine prípadov nemal počet neurónov zásadný vplyv na počet cyklov a priemer bol výrazne vyšší len v jednej pätici testov, čo zvýšilo aj celkový priemer.

- **Vplyv počtu záznamov –**

Počet záznamov	Priemer počtu cyklov	Priemerný čas
4097	90,9	18,73
2050	115	10,86

Tabuľka 7.3 Vplyv počtu neurónov

S klesajúcim počtom záznamov sa zvýšil počet prechodov sieťou, keďže na naučenie siete bolo v jednom cykle menej výpočtov. Čas sa naopak zvýšil pri väčšom počte záznamov, aj keď sieť potrebovala na naučenie menej cyklov, čo je následok viacerých výpočtov v jednom cykle.

- **Vplyv počtu atribútov -**

Počet atribútov	Priemer počtu cyklov	Priemerný čas
23	133,4	21,15
11	74,5	8,45

Tabuľka 7.4 Vplyv počtu neurónov

Počet atribútov sa ukázal ako parameter , ktorý má zo sledovanýchparmetrov najväčší vplyv na počet cyklov ako aj na čas. Rozdiel v čase je takmer rovnaký ako pri zmene počtu neurónov, avšak počet cyklov sa zo zvýšením počtu atribútov výrazne zvýšil. Ako som už spomínal iba jediný atribút je pre výslednú klasifikáciu do triedy dôležitý a nárastom počtu atribútov výrazne narástol aj počet prechodov sieťou aby sa tento fakt naučila.

Ako sa ukázalo vplyv náhodného nastavenia váh a biasov je zjavný, ako aj vplyv nárastu ktoréhokoľvek zo sledovaných parametrov na čas. Počet prechodov sieťou úzko súvisí z každým z týchto parametrov, keďže s narastajúcim počtom neurónov narastá aj zložitosť šírenia chybymedzi nimi. Keďže zvýšenie počtu atribútov má za následok najväčší nárast počtu neurónov aj rozdiel počtu prechodov sieťou je najväší.

8 Záver

Cieľom tohto projektu bolo dôkladné preštudovanie problematiky dolovania z dát a neurónových sietí a po tejto príprave doimplementovať dolovací modul. Moja príprava mala niekoľko fáz, na začiatku som musel naštudovať nevyhnutnú teóriu dolovania dát z databáz ako aj samotnú implementáciu projektu Dataminer. Následne som musel naštudovať problematiku neurónových sietí a ich využitia v dolovaní z dát. Cieľom projektu bolo nie len implementovanie modulu ale aj prezentácia výsledkov, čoho som dosiahol pomocou grafu a hlavne možnosťou aplikovať importované dáta na naučenú neurónovú sieť.

Na záver tejto bakalárskej práce by som hlavne chcel rozobrať jej prínos ako pre aplikáciu Dataminer tak aj pre mňa osobne. Z pohľadu aplikácie je to ďalšie jej rozšírenie prispievajúce k jej komplexnosti v riešení dolovacích úloh o ďalší dolovací modul, ktorý vie elegantne riešiť aj nelineárne problémy.

Z môjho osobného pohľadu mi táto práca dala veľa vedomostí, ktoré mi budú užitočné v ďalšom štúdiu ako aj v praxi. Či už ide o spomínanú problematiku neurónových sietí a ich aplikácie v dolovaní z dát ale aj skúsenosti s prácov na rozpracovanom projekte.

Hlavným možným rozšírením je ukladanie už natrénovanej siete do DMSL, ktoré som z časových dôvodov nedokončil a tak som implementáciu ukladania úplne z práce vylúčil. Ďalšou možnosťou rozšírenia je postupné trénovanie a testovanie siete na množine trénovacích a testovacích dát. Ďalej by bolo možné doimplementovať vizuálny model siete pomocou grafu. Ďalšie rozšírenie by bolo možné doimplementovať do modulu rozhranie pre voľbu neurónovej siete a aj ďalšie neurónové siete napr. Kohenovu sieť.

Literatúra

[1] Zendulka, J.; Bartík, V.; Lukáš, R.; aj.: Získávání znalostí z databází – studijní opora. FIT VUT v Brne, Brno, 2006.

[2] Han, J.; Kamber, M.: Data Mining: Concepts and Techniques. Elsevier Inc., druhé vydanie, 2006, ISBN 978-1-55860-901-3, 770 s.

[3] Wikipedia: Perceptron – Wikipedia, The Free Encyclopedia. 2010, [Online; navštívené 15. 1. 2010]. URL http://en.wikipedia.org/wiki/Perceptron

[4] Gopalan, S: Data Mining. PHI Learning Pvt. Ltd., ISBN 978-81-203-3812-8.

[5] Lukaš, R: Klasifikace a predikce-slajdy k předmětu ZZN. FIT VUT v Brně.

[6] Kotásek P.: DMSL: The Data Mining Specification Language, Disertační práce, FIT VUT v Brně, 2003. URL http://www.fit.vutbr.cz/research/view_pub.php?id=7348

[7] NetBeans, [Online; navštívené 15. 1. 2010], URL: http://www.netbeans.org/features/index.html

[8] Krásny, M: Systém pro dolování z dat v prostředí Oracle, diplomová prace, Brno, FIT VUT v Brně, 2008.

[9] Šebek, M: Rozšíření funkcionality systému pro dolování dat na platformě NetBeans, diplomová práce, Brno, FIT VUT v Brně, 2009.

[10] Cole, G: Backprop1, [Online; navštívené 15. 1. 2010] . URL: http://backprop1.sourceforge.net/

[11] Welcome To JFreeChart!. JFreeChart, [Online; navštívené 15. 1. 2010]. URL: http://www.jfree.org/jfreechart/

Zoznam príloh

Príloha A: Dátový disk CD