Mendelova univerzita v Brně
Provozně ekonomická fakulta

Implementace metod pro automatické dolování dat

Bakalářská práce

Vedoucí práce:
prof. RNDr. Ing. Jiří Šťastný, CSc.

Peter Smatana

Brno 2015

Neexistují slova, superlativy nebo výroky moudrých lidí, které by patřičně vyjádřili můj vděk. Tuto práci bych rád věnoval svým rodičům. Úžasným lidem, kteří pro mě udělali mnohé a já jsem ji nezkonale vděčný. :-) Poděkovat bych chtěl panu profesorovi Jiřímu Šťastnému, mimojiné za vedení této práce. Nakonec bych rád poděkoval Odinovi Popelkovi za dobré rady do profesního života.

Čestné prohlášení

Prohlašuji, že jsem tuto práci: **Implementace metod pro automatické dolování dat**
vypracoval samostatně a veškeré použité prameny a informace jsou uvedeny v seznamu použité literatury. Souhlasím, aby moje práce byla zveřejněna v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů, a v souladu s platnou *Směrnicí o zveřejňování vysokoškolských závěrečných prací.*
Jsem si vědom, že se na moji práci vztahuje zákon č. 121/2000 Sb., autorský zákon, a že Mendelova univerzita v Brně má právo na uzavření licenční smlouvy a užití této práce jako školního díla podle § 60 odst. 1 Autorského zákona.
Dále se zavazuji, že před sepsáním licenční smlouvy o využití díla jinou osobou (subjektem) si vyžádám písemné stanovisko univerzity o tom, že předmětná licenční smlouva není v rozporu s oprávněnými zájmy univerzity, a zavazuji se uhradit případný příspěvek na úhradu nákladů spojených se vznikem díla, a to až do jejich skutečné výše.

V Brně dne 19. května 2015 .....

Abstrakt

Práce popisuje proces dolování dat na úloze Give Me Some Credit. Práce provádí čtenáře postupně celým procesem dolování dat. V úvodu je seznámení s problematikou dolování dat a dolovanými daty. Dále je představen vlastní nástroj na čištění a přípavu dat. Na závěr je provedeno modelování a zhodnocení výsledů.

Klíčová slova: Data mining, R, IBM SPSS Modeler, Credit scoring, Java

Abstract

This thesis describes the process of data mining on exercise Give Me Some Credit. Work carry out reader through the entire process of data mining. In the introduction is explain issue what is data mining and what data will be mine. Further is introduced tool for cleaning and preparing data. At the end modeling is done and evaluation of the results.

Key words: Data mining, R, IBM SPSS Modeler, Credit scoring, Java

Obsah

1 Úvod, cíl práce a metodika

1.1 Úvod

Dolování znalostí z dat je způsob, jak najít v datech skryté nebo neznámé informace. Získávání těchto skrytých znalostí dovoluje dělat lepší rozhodnutí nikoliv na základě doměnek nebo intuice, ale na základě skutečných událostí, které data představují.

1.2 Cíl práce

Na základě nastudované literatury a pochopení dostupných materiálů jsem popsal analyzovaná data fiktivní finanční společnosti. Ve fázi přípravy a čištění dat budu programovat vlastní nástroj, který obslouží všechny požadavky na předzpracování dat. Na závěr provedu klasifikaci klientů, kteří jsou nebo nejsou schopni plnit své závazky.

1.3 Metodika

V kapitole 2 *Současný stav* popisuji všechny nezbytné teoretické základy, k pochopení problematiky dolování dat. V kapitole 3 *Návrh řešení problému* popisuji s jakými daty pracuji a jaká jsou specifika dat týkající se finanční problematiky. Dále popisuji vlastní nástroj předzpracování dat. V poslední kapitole 4 *Řešení problému* rozebírám samotné dolování dat, hrozby které z činnosti plynou a jejich řešení. V závěru hodnotím dosažené výsledky a diskutuji možnosti další práce. Součástí práce je i příloha s obrázky, tabulkou a seznamem literatury, ze které jsem čerpal a kterou doporučuji zájemcům o hlubší studium dolování dat.

2 Současný stav

2.1 Co je dolování dat

Podle (Larose, 2005)[2] je „data mining proces objevování smysluplných souvislostí, vzorů nebo trendů z velkého množství dat uložených v nějaké bázi. Toho je možné dosáhnout použitím statistických a matematických metod.“ Publikace (Berry, 2004)[7-8] podobně jako (Larose, 2005)[2] nejřív uvádí, že dolování dat je „prohledávání a analýza velkého množství dat, díky kterému se objeví užitečné vzorce nebo pravidla“. Následně pojem zužuje na oblasti, kterými se kniha primárně zabývá. „Dolování dat pomáhá společnostem dělat lepší rozhodnutí v oblastech marketingu, prodeje nebo zlepšování vztahů se zákazníky“. Z těchto myšlenek mohu říct, že dolování dat nebo-li data mining je analytická metodologie získávání netriviálních, skrytých a potenciálně užitečných informací.

2.2 Proč používat data mining a proč data mining automatizovat

Vznik datové analýzy se datuje od 80 až 90 lét, kdy byly k dispozici databázové stroje umožňující pohodlné skladování a manipulaci s daty. Současně s databázemi bylo třeba nějakého dostupného výpočetního výkonu, který tehdejší počítače již byly schopny nabídnout. Historie dolování znalostí z dat sahá více do historie, kdy výhradně muži pracující v bankách a finančních institucích museli narukovat do války. Z těchto důvodů vymýšleli automatické procesy, které klientům schvalovaly nebo zamítaly finanční produkty.

Dnes se data mining snaží propojovat pohled statistiky a matematiky na data z jedné strany. Z druhé strany si vypomáhá strojovým učením nebo analýzou časových řad. Data mining přináší ucelené metody, jak znalosti z dat získat, takže analytik nemusí jednoduše řečeno znovu objevovat „kolo“.

Automatizovaného přístupu se používá proto že dat, které je třeba nutné zpracovat je takové množství, že není v silách člověka je relevantně ručně zpracovat. Navíc je pochopitelné, že člověk podléhá různým náladám nebo únavě, které se podepisují na výsledku práce. Automatickými postupy si můžeme dovolit zpracovávat velké množství jednotlivých pozorování, které mají mnoho vlastností a atributů v relativně krátkém čase.

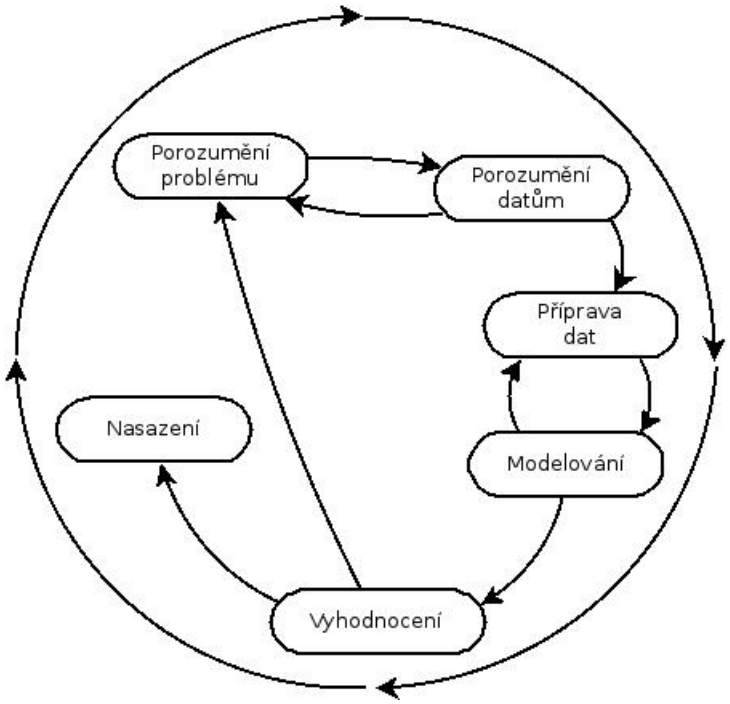
2.3 Typy úloh dobývání znalostí

V obecné rovině dolování dat rozlišujeme tyto úlohy.

- **Klasifikace** spočívá v přiřazení nějakého pozorování do předem známé třídy. Klasifikace v praxi funguje tak, že existuje sada pozorování již přiřazených do tříd (viz kapitola 2.5). Nad trénovací množinou se vytvoří model, který je schopen neznámá pozorování zařazovat do již definovaných tříd. (Berry, 2004)[8-9]
- **Odhady** se od klasifikace liší tím, že klasifikace přiřazuje pozorování do určité diskrétně dané třídy. Například „muž“, „žena“ nebo „savec“, „ryba“, „pták“, „plaz“. Odhad pracuje se spojitou veličinou. Přiřazuje například klientovi finanční společnosti koeficient 0 až 1 podle toho, kolik klient instituci svěřuje finančních prostředků. (Berry, 2004)[9-10]
- **Předpovědi** jsou podle (Berry, 2004)[10-11] nebo (Larose, 2005)[13] stejné jako klasifikace nebo odhady s tím rozdílem, že výsledky, které hledám nejsou známé, leží v budoucnu. Tradiční úlohou může být predikce vývoje ceny koruny vůči americkému dolaru.
- **Asociace** odpovídají na otázky týkající se vztahu věcí v klientově nákupním košíku. Jinými slovy asociační pravidla ukazují, jak moc spolu souvisí to, že si klient koupil například pivo, dětské pleny a chleba. (Berry, 2004)[11]
- **Shlukování** je podle (Berry, 2004)[11] a (Larose, 2005)[16-17] velmi podobné klasifikaci, jelikož u shlukování dáváme do jedné skupiny entity, které jsou si nějak podobné. Rozdíl oproti klasifikaci je ten, že shlukujeme do skupin, které nejsou dopředu známé.
- **Popisování** nebo-li deskripce jsou metody, jak popsat a pochopit data. Často se stává, že jsou data spojovaná s veřejně dostupnými informacemi jako je například počasí. (Larose, 2005)[11]

2.4 CRISP-DM

Je zkratkou pro *The Cross–Industry Standard Process for Data Mining* a označuje doporučovaný postup šesti kroků, které budou-li dodrženy, tak ze syrových dat budou vydestilovány dosud neznámé a potenciálně užitečné znalosti. Každý krok se skládá z dalších obecných doporučení.



Obrázek č. 1: Cyklus fází CRISP-DM.

Porozumění problému

Porozumění problému je první fáze metodiky CRISP–DM. Někdy je tento krok označován jako „definování cílů“, což znamená ujasnit si, co chci dolováním dat dosáhnout. (Larose, 2005) V případě finanční instituce to může být snaha zvýšit počet klientů, kteří si zvolí další produkt. Nebo naopak se může jednat o snahu snížení podvodného chování svých klientů. V praxi společnosti řeší kolik zaměstnanců bude do projektu zapojeno, a jak dlouho bude realizace trvat, než budou dosaženy výsledky. Z pohledu lidských kapacit se jedná o zapojení IT odborníků minimálně z oblasti databází a datových skladů. Dále jsou to samotní analytici nebo právníci, kteří zkoumají, nejsou-li kroky v rozporu s právním řádem. Kupříkladu problematická manipulace s osobními údaji. Posledním článkem lidského řetězce může být marketingové oddělení, které klientům představuje změny v produktech nebo produkty nové.

Porozumění datům

Porozumění datům je klíčová fáze, kde zkoumám jaká data mám k dispozici. Co přesně atributy znamenají a jestli se jejich chápání v průběhu času nějak měnilo. Potřebuji vědět, jak jsem data získal, došlo-li v čase ke změně měřítka nebo rozsahu hodnot, jakou data mají výpovědní hodnotu a bylo-li v průběhu času sběru kontrolována jejich kvalita. Například bylo-li telefoní číslo na klienta ověřeno. Je třeba věnovat pozornost číselníkům a jejich změnám, důvodu vzniku nových tabulek nebo

sloupců a jak se přistoupilo k doplnění dat tak, aby například nový sloupec obsahoval kompletní data.

Příprava dat

Příprava dat nebo také čištění dat je moment, kdy data dostávám z různých formátů do jedné tabulky. Zdroje dat mohou být tradiční databáze, datové sklady, soubory pocházející z tabulkových procesorů jako například Microsoft Excel, OpenOffice.org Calc nebo LibreOffice Calc. Zdrojem dat mohou být i textové soubory nebo soubory s vnitřní strukturou jako například CSV[1], XML[2] nebo JSON[3]. Pokud mám k dispozici všechny zdroje dat, mohu se rozhodovat, která data budou použita pro analýzu. Kritérií k výběru dat je několik. Data musí být „čistá“, to znamená, že se v datech nemohou objevovat chybějící hodnoty. Na odhadnutí chybějících hodnot existují nejrůznější metody. Mohou to být jednoduché metody, kdy se za chybějící hodnotu dosadí například aritmetický průměr daného atributu. Mohou být ale i metody složitější, založené na odhadech, modelování pravděpodobnosti a podobných statistických přístupech. (Little, 2002) Často se stává, že data jsou k dispozici ve formě, která explicitně nepředpokládala jejich dolování. Proto často dochází k tvorbě nových nebo odvozených atributů a nebo transformaci stávajících atributů.

Modelování

Modelování je kýžená část metodologie CRISP–DM. Připravená a vyčištěná data se rozdělují na tři podmnožiny, kterým se říká testovací, trénovací a validační množina (viz kapitola 3.5). Tyto podmnožiny dat jsou předkládána dolovacímu algoritmu, jehož výstupem je jeden model nebo skupina modelů. Existují přístupy, kdy se vygeneruje jeden model a při dalším generování se vychází z tohoto již vygenerovaného modelu. Snahou je nově vzikající model vylepšit na základě starého modelu.

U modelování je důležité zvolit cílovou proměnnou. Je to taková proměnná, která zjednodušeně vyjadřuje výstup modelování, nebo-li to, co má být výstupem klasifikace, odhadu, predikce nebo shlukování. (Berry, 2004)[56-58] Aby model mohl pracovat a na výstupu vrátit cílovou proměnnou, je třeba modelu na vstupu specifikovat prediktivní proměnnou. Typicky prediktivních proměnných je několik. Můžeme si je představit jako sloupce v tabulce tabulkového procesoru nebo databáze či datového skladu. (Larose, 2005), (Berry, 2004)

[1] Comma-separated values
[2] Extensible Markup Language
[3] JavaScript Object Notation

Nasazení

Nasazení data miningových modelů je proces, který se řeší v praxi a nejčastěji zahruje daleko víc kroků, než jen samotné převedení modelu do reálného prostředí. První krok, který musí být známý předem je schopnost model v produkčním prostředí nasadit. Pokud bude model tvořený složitou neuronovou sítí, bude jeho implementace podstatně náročnější než model založený na lineární regresi. Nasazení obsahuje také průběžnou kontrolu, plní-li model požadavky či nikoli. Nasazení modelu se musí provádět také v kontextu technologií, které jsou k dispozici a které podnik nebo firma používá. Je to střet, kdy například datový analytik používá data miningový software IBM SPSS Statistics pro tvorbu modelů, ale nasazení modelu v produkčním prostředí bude implementováno nad databází Oracle 12c. Může nastat situace, že každý z těchto systémů pracuje odlišně s desetinným rozvojem.

SEMMA

Je metodika pocházející od společnosti SAS. Jednotlivá písmena jsou zkratkou pro doporučené činnosti datové analýzy.

1. **Sample** – vzorek z celého souboru dat je k dispozici v jedné nebo více tabulek. Vzorek dat by měl být náhodný, aby nedošlo ke zkreslujícímu pohledu na celý soubor dat.
2. **Explore** – prozkoumávání dat za účelem nalezení očekávaných vztahů nebo naopak neočekávaných trendů a přehlédnutých skutečností. Hledají se ovšem v datech i anomálie nebo z podstaty věci chybná pozorování.
3. **Modify** – z jednoduchých závěrů, které vyplývají z předchozího kroku „prozkoumávání dat“ se data pro další práci transformují, vytvářejí se nové proměnné a nevyhovující atributy se zahazují.
4. **Model** – trénování modelu pomocí algoritmů dostupných v SAS software. Uživatel může použít různé druhy regresí, rozhodovacích stromů nebo neuronových sítí.
5. **Access** – zpřístupnění modelu za použití různých grafů, lift křivek, tabulek vyjadřující absolutní čísla nebo procenta.

2.5 Strojové učení

Strojové učení je schopnost výpočetní techniky „učit se“ na základě vstupů, nejčastěji z dat. Algoritmy strojového učení fungují tak, že na vstupu mají data, podle kterých se „učí“. Příkladem může být úloha rozpoznat na obrázcích kočku nebo počítač. Těmto datům říkáme *trénovací data*. Na základě trénovacích dat algoritmy vytvářejí model, který reprezentuje naučenou znalost. V tomto případě algoritmus na vstupu dostane tisíc obrázků nejrůznějších koček a počítačů s označením co je

kočka a co je počítač. Vytvořenému modelu jsou předkládána neznámá data a model musí určit, jaké povahy data jsou. Těmto datům říkáme *testovací data*. Existují také *validační data*, která ověřují správnost chování modelu, potažmo algoritmu. (Smola, 2008)[3-13] (Witten, 2011)

Učení s učitelem

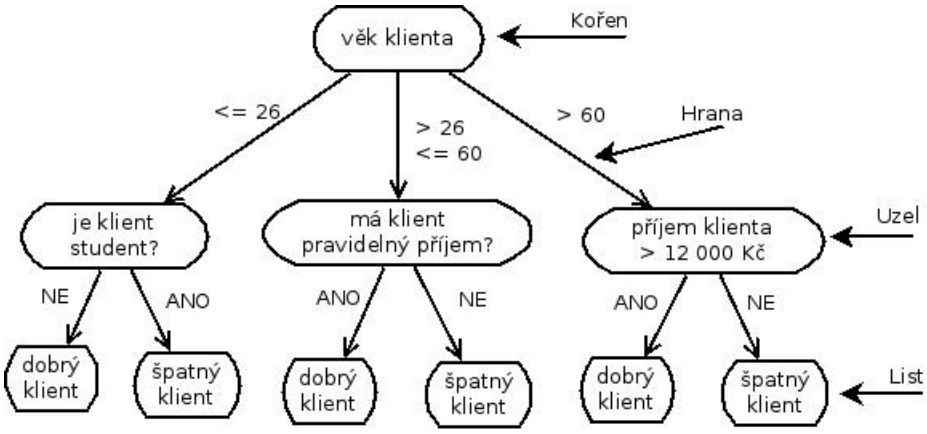
Učení s učitelem je takové strojové učení, které pracuje s označenými daty. Příkladem může posloužit přiřazování novinových článků do kategorií „politika“, „ekonomie“ a „sport“. (Smola, 2008) (Witten, 2011)

Učení bez učitele

Učení bez učitele je pravým opakem učení s učitelem, kde předem nejsou známé kategorie. Tyto kategorie algoritmus sám odvozuje. (Smola, 2008) (Witten, 2011)

2.6 Rozhodovací stromy

Rozhodovací stromy jsou oblíbeným nástrojem interpretace dat. Je to dáno grafickou reprezentací, ve které se člověk snadno orientuje. Z teorie grafů, rozhodovací stromy mohou být chápány jako neorientované souvislé grafy, které neobsahují kružnici.



Obrázek č. 2: Příklad rozhodovacího stromu.

Rozhodovací stromy obsahují tyto komponenty:

- **Uzel** je komponenta, která obsahuje data podléhající určitému rozhodovacímu pravidlu. Mohu například požadovat uzel obsahující množinu klientů, jejichž věk je menší nebo roven dvaceti šesti.
- **Kořen** je uzel, který obsahuje vstupní nerozdělená data. Do kořene nevstupuje žádná hrana.

- **Hrana** je spojnice dvou uzlů a vyjadřuje rozhodovací pravidlo. Takové pravidlo mohou být klienti, jejichž příjem je větší než dvanáct tisíc korun. Hranou potom je „ano“ a „ne“ což vytváří nový uzel, odpovídající tomuto pravidlu.
- **List** je uzel, který má pouze rodiče a nemá následovníka. List představuje cílovou klasifikovanou nebo predikovanou třídu.
- **Sled** je taková posloupnost uzlů u a hran h, že platí $\langle u_0, h_1, u_1, h_2, \ldots, h_{n-1}, u_{n-1}, h_n, u_n \rangle$. u_0 je kořen a u_n je list. Sled u rozhodovacích stromů chápeme jako soustavu pravidel, které vytvoří cílovou klasifikovanou nebo predikovanou třídu.
- **Hloubka uzlu** je délka cesty od kořene k danému uzlu.

Problémy rozhodovacích stromů

Problém rozhodovacích stromů je výběr správných atributů, které budou rozdělovat data. Může se stát, že data budou přísně rozdělována do malých množin a tím vznikne velký strom s mnoha listy. Tyto listy se často nachází ve velké hloubce a rozhodovací strom se stane komplikovaným vzhledem k pravidlům, které určují list. Na druhou stranu velmi rozrostlý strom s mnoha listy má tendenci mít listy s vysokou čistotou. Pro kasifikační úlohy to znamená, že v listu se nachází převážně prvky jedné třídy. Naopak strom, který má volněji nastaveno rozdělování dat do tříd má vyšší míru chyby. To znamená, že v listech se objevují data z více jak jedné třídy. Měřítkem čistoty listů může být entroie. (Han, 2012)[336]

3 Návrh řešení problému

3.1 Popis dat

Data, která jsem si vybral k analýze se týkají finanční problematiky. Data pochází z portálu `www.kaggle.com`. Kaggle je portál, který obsahuje úlohy týkající se data miningu v příjemné podobě pro studenty a zájemce o datovou analytiku. Úloha se jmenuje Give Me Some Credit a obsahuje dva soubory a popis dat. Jeden soubor obsahuje trénovací množinu dat, druhý obsahuje testovací data. Na tento typ dat se dají uplatnit techniky, kterým se souhrně říká credit scoring (viz kapitola 4.2). Tabulka 1 popisuje změnu názvů z originálního data setu. Důvodem je lepší uchopitelnost kratšího názvu než delšího a komplikovaného. Data jsou záznamy o klientech fiktivní finanční společnosti. Data přímo neodrážejí historii chování klienta, pouze historii zjednodušují. Atributy uchovávané o klientovi přibližuje tabulka 2.

Název sloupce z Kaggle	**Vlastní název sloupce**
SeriousDlqin2yrs	pastDeliquency
RevolvingUtilizationOfUnsecuredLines	wholeDeliquency
Age	age
NumberOfTime30-59DaysPastDueNotWorse	x3059
DebtRatio	debtRatio
MonthlyIncome	income
NumberOfOpenCreditLinesAndLoans	openLoans
NumberOfTimes90DaysLate	x90
NumberRealEstateLoansOrLines	realLoans
NumberOfTime60-89DaysPastDueNotWorse	x6089
NumberOfDependents	dependents

Tabulka č. 1: Přejmenování názvů atributů.

Název atributu	Popis atributu	Typ atributu
pastDeliquency	Nesplatila-li osoba svoje závazky po 90 dnech od splatnosti. Tento atribut je cílová proměnná.	logická hodnota
wholeDeliquency	Zůstatek na kreditních kartách a ostatních půjčkách kromě hypoték.	procento
age	Věk klienta.	přirozené číslo
debtRatio	Součet splátek, alimentů, životních nákldů a vydělený hrubým příjmem.	procento
income	Měsíční příjem klienta.	reálné číslo
openLoans	Počet otevřených půjček nebo úvěrů.	přirozené číslo
realLoans	Počet hypoték a úvěrů na nemovitosti.	přirozené číslo
x3059	Počet závazků, které klient nesplatil ani po 30 až 59 dnech po splatnosti.	přirozené číslo
x6089	Počet závazků, které klient nesplatil ani po 60 až 89 dnech po splatnosti.	přirozené číslo
x90	Počet závazků, které klient nesplatil ani po 90 dnech po splatnosti.	přirozené číslo
dependents	Počet rodinných příslušníků závislých na klientovi. Například manžel, manželka nebo děti.	přirozené číslo

Tabulka č. 2: Popis atributů klienta.

3.2 Credit scoring

Credit scoring je metodika finančních společností a bank, která popisuje jak svým klientům například půjčovat finanční prostředky nebo jak odhalit problematické žadatele o úvěr. Takové půjčování je spjaté s riziky, že klient svoje závazky nebude plnit. Momentem zautomatizování credit scoringového modelu byly ve dvacátém století dvě světové války. Do té doby posuzování a schvalování úvěrů probíhalo individuálně. Bylo defakto standardem, že ve finančních institucích a bankách pracovali výhradně muži. Jejich odchodem do armády vznikla potřeba credit scoringové metody automatizovat.

Strategií jak vytvářet credit scoringové modely existuje několik. (Siddiqi, 2006)[22-25]

- Snížení špatných půjček, bankrotu nebo podvodů.
- Snížení pohledávek, které již není možné vymáhat.
- Zvýšení ziskovosti podniku.

Scorecard

Pojem scorecard nebo také scoringové karty se používá běžně v praxi. Pojem zahruje definici cíle a postup, jak ho dosáhnout. Cíl může být zvyšování zisku nebo snížení počtu podvodů. Postup jak cíle dosáhnout je například neposkytovat finanční prostředky klientům mladším dvaceti šesti let. Forma scorecard musí být jednoduše interpretovatelná pro všechny zaměstnance finanční instituce, protože v ní pracuje mnoho lidí, kteří nemusí mít znalosti statistiky nebo data miningu. (Siddiqi, 2006)[26-27] V mém případě bude scoringová karta jednoduché pravidlo, je-li klient z pohledu splácení pohledávek vůči finanční instituci *dobrý* nebo *špatný*.

Vylučování klientů

Existují typy klientů, se kterými se při credit scoringovém procesu setkáváme každý den, a naopak existují tací klienti, kteří vykazují abnormální hodnoty. První skupina klientů jsou ti, které chceme mít v trénovací množině. Druhá skupina klientů má například nadstandartní příjem, jsou klasifikováni jako VIP[1] nebo se jedná o účty zaměstnanců finanční instituce. Druhou skupinu klientů v trénovací množině nechceme, protože by pokřivil model klasických klientů. (Siddiqi, 2006)[31-32]

Definice dobrého a špatného klienta

Segmentovat klienty na dobré a špatné je fundamentální problém, jelikož na zakladě této informace se klientovi schválí nebo neschválí čerpání finančních prostředků. Častým parametrem schválení nebo neschválení úvěru je 30/60/90 nebo-li počet delikvencí po třiceti, šedesáti respektive devadesáti dnech po splatnosti pohledávky.

3.3 Čištění dat

Problematika čištění dat

Existuje mnoho nástrojů na čištění dat, podchycení chybějících hodnot, transformaci sloupců nebo celých dat. Tyto nástroje jsou vhodné pro rychlé zvládnutí nějakého problému. V praxi se ale setkáme s potřebou, kdy různé datové transformace potřebujeme provádět opakovaně a automatizovaně. Je rozhodně snažší a jednodušší používat nástroje, které zde již existují a jsou u datových analytiků oblíbené. Problém nastává ve chvíli, kdy s daty potřebujeme provádět velmi komplikované operace, které jsou ovlivněné nějakou další skutečností. Například načítat

[1] very important person – velmi důležitá osoba

data z databáze a zároveň z XML souborů a transformovat je podle pravidel, která se nacházejí na vzdáleném serveru ve formátu CSV. Aby proces probíhal automatizovaně, je třeba existující nástroj ovládat například pomocí API[1] nebo pomocí nějakého vestavěného programovatelného systému. Pokud tato funkcionalita chybí, může to znamenat zavrhnutí používání takového nástroje.

Na základě těchto rozvah jsem vytvořil v programovacím jazyku Java vlastní nástroj. Tento nástroj jsem nazval *DataFarmer*. Název programu má evokovat obhospodařovávání nebo obdělávání dat. Program data načítá a ukládá ve formátu, který si uživatel zvolí. DataFarmer umí podchytit a zpracovat chybějící hodnoty, a to ve dvou stavech, načíst syrová data a vrátit klienty, kteří mají nebo nemají chybějící hodnoty. DataFarmer je schopen z načtených dat vybírat taková data, která odpovídají nějakým požadavkům. Například vybrat všechny klienty s věkem v určitém intervalu a zároveň s počtem na nich závislých lidí, jež je roven jedné. DataFarmer je schopen z dat vybrat náhodnou podmnožinu, určenou procentuelním podílem nebo vytvářet nové sloupce.

3.4 Program DataFarmer

DataFarmer jsem primárně navrhnul se značným důrazem na rozšíření. Ústřední jednotkou je třída `DataFarmer`, která používá druhou nejdůležitější třídu `ETL`. Třída `ETL` představuje fasádu (viz kapitola 4.5) pro třídy `CreateColumn`, `DataSubset`, `IntervalValues` a `MissingValues`. Tyto třídy vytváří nové sloupce, vrací podmožinu dat nebo data jejichž atributy patří do zadaného intervalu. Třída `ETL` zapouzdřuje funkcionalitu vytváření nových sloupců, vracení dat omezených nějakým pravidlem nebo zpracování chybějících hodnot. Pro náčítání dat z různých zrojů zde existuje opět fasáda v podobě třídy `DataIO`, která enakpsuluje třídy `DatabaseHandler` a `FileHandler`. Tyto třídy mohou data načítat ze souboru nebo z databáze. Pokud si uživatel bude přát implementovat vlastní funkcionalitu načítání dat, musí jeho nová třída implementovat rozhraní `IDataIO`. Kompletní přehled tříd ilustruje tabulka 3. Vybrané třídy jako například `ETL` nebo `ApplicationSettings` jsou realizovány podle návrhového vzoru jedináček (viz kapitola 4.5). Zjednodušený diagram tříd (Hamilton, 2006)[93-120] ilustruje obrázek 3. Jsou na něm vyjádřeny pouze vztahy mezi třídami. Třídy spolu s atributy a metodami jsou uvedeny v příloze.

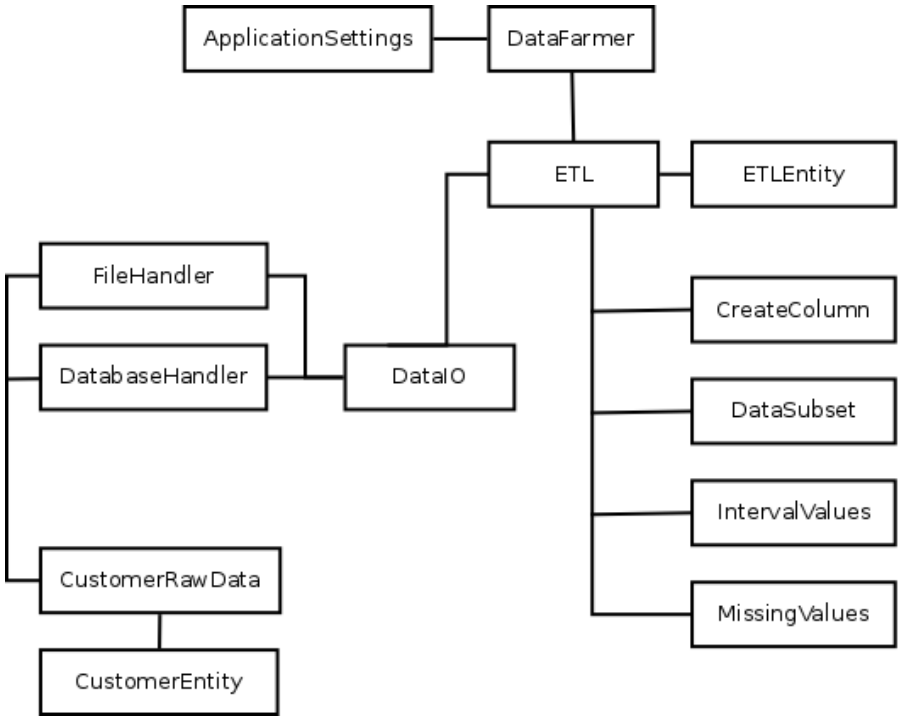
Při práci s programem DataFarmer uživatel používá třídu `ETL`. Na začátku manipulace s daty uživatel specifikuje, kde se mají načítat. Následně určí, že chce pracovat s daty, které nemají chybějící hodnoty. Potom intuitivně přistupuje k atributům a určuje jak bude vypadat výsledná podmnožina. Toto ilustruje zdrojový kód 1, kde na řádku 3 uživatel načte všechna data, pro která platí, že atribut *realLoans* bude větší nebo roven 3. Na řádku 5 uživatel vybírá ty klienty, jejichž atribut *age* je menší než 35. Tento postup je vyjádřen aktivity diagramem (Hamilton, 2006)[62-91] na obrázku 4.

[1] Application Programming Interface

```
1 ETL.loadData(pathToLoad);
2 ETL.getCustomersWithoutMissingValues();
3 ETL.getCustomersWithAttributeGreaterThanEqual("realLoans", 3.0);
4 ETL.getCustomersWithAttributeEqual("x3059", 0.0);
5 ETL.getCustomersWithAttributeLessThan("age", 35.0);
6 ETL.saveData(pathToSave);
```

Zdrojový kód č. 1: Příklad práce s programem DataFarmer.

Obrázek č. 3: Diagram aktivit představující zpracování dat v programu DataFarmer.



Obrázek č. 4: Zjednodušený diagram tříd.

Název třídy	Popis třídy
`CustomerEntity`	Jednoduchá třída obsahující atributy, které reflektují všechny atributy uchovávané u klientů.
`CustomerRawData`	Třída obsahuje pouze načtená syrová data o zákaznících, která se dále zpracovávají.
`DataIO`	Třída, která zapouzdřuje načítání dat z různých zdrojů.
`DatabaseHandler`	Připravená třída pro načítání a ukládání dat do databáze.
`FileHandler`	Třída, která načítá a ukládá data do textového souboru.
`IDataIO`	Rozhraní, jež musí implementovat každá třída, která do aplikace načítá data a opět je ukládá.
`ETL`	Fasáda, která zpřístupňuje všechnu funkcionalitu, kterou je možné aplikovat na data.
`ETLEntity`	Všechny transformace, filtrace a další operace prováděné nad daty jsou jednotně ukládané spolu s přesným popisem jaká operace se provedla.
`CreateColumn`	Třída vytváří nové sloupce.
`DataSubset`	Třída vrací náhodnou podmnožinu dat v procentuálním nebo absolutním vyjadření.
`InternalValues`	Třída vrací podmnožinu dat na základě názvu atributu a hodnoty, která podmnožinu ohraničuje.
`MissingValues`	Třída vrací data, která obsahují nebo neobsahují chybějící hodnoty.
`ApplicationSettings`	Tato třída obsahuje kompletní nastavení aplikace agregovaného do jednoho místa.
`DataHandling`	Hlavní třída aplikace, která celý program spouští.

Tabulka č. 3: Popis všech tříd programu DataFarmer.

3.5 Návrhové vzory použité v nástroji DataFarmer

Fasáda

Návrhový vzor fasáda se používá ve chvíli, kdy je třeba zjednodušit chování komplexního nebo složitého systému. Fasáda slouží k nahrazení mnoha dílčích systémů sjednocených rozhraním, které tyto systémy ovládá. Jinými slovy, programátor definuje rozhraní vyšší úrovně, které usnadňuje používání podsystémů a zároveň odstiňuje programátora od používání mnoha tříd na místo jednoho styčného bodu. (Pecinovský, 2007)[255-260]

Jedináček

Návrhový vzor jedináček, často známý pod názvem „singleton“ je jednoduchý vzor, který zabezpečí, aby u požadované třídy došlo k vytvoření právě jedné instance.

Návrhový vzor singleton se používá v momentě, kdy není třeba, aby při běhu programu bylo vytvořeno více instancí dané třídy nebo kdy si toto chování programátor explicitně nepřeje. Například třída `ApplicationSettings` uchovává nastavení aplikace a není důvod, aby v systému existovalo více instancí, když nastavení je stále jedno a to samé. (Pecinovský, 2007)[107-121]

3.6 Testování

V průběhu programování nástroje DataFarmer jsem si pro vlastní kontrolu správnosti funkcionalit programoval testy. Nepoužil jsem žádný framework. Testy jsem koncipoval jako jednoduché scénaře, kde na začátku vytvořím počáteční podmínky, spustím určitou testovanou funkcionalitu a kontroluji výsledky. Příklad jednoduchého testu ukazuje zdrojový kód 2, kde se na řádku 2 načítám testovací data. Na řádku 5 vyberu klienty bez chybějících hodnot a na dalším řádku vyberu všechny klienty, kteří mají jednoho člověka, který je na něm závislý (viz tabulka 2). Na závěr testuji shodu mezi počtem vyfiltrovaných klientů a skutečným počtem, kteří scénáři vyhovují.

```
1  private void test_equal() {
2          ETL.getInstance().loadData(MainTest.getGoodData());
3          IntervalValues internalValues = new IntervalValues();
4          MissingValues missingValues = new MissingValues();
5          ETLEntity etlEntity = missingValues.
               getAllCustomersWhoHasNotMissingValue(ETL.getActualData());
6          ETLEntity testETLEntity = internalValues.returnAttributeEqual(
               etlEntity.getEtlState(), "dependents", 1.0);
7          if (testETLEntity.getEtlState().size() == 2) {
8                  MainTest.printPassTest();
9          } else {
10                 MainTest.printFailTest("test_equal");
11         }
12 }
```

Zdrojový kód č. 2: Testový scénář programu DataFarmer.

4 Řešení problému

4.1 Korelace atributů

První informace, kterou potřebuji znát, je jak spolu všechny atributy, které se u klienta evidují spolu suvisejí nebo-li korelují. Korelace je vztah mezi dvěma veličinami. Pokud se jedna veličina mění, potom se korelativně mění i druhá veličina. Podle tabulky 9, která se nachází v příloze je patrné, že nejvíc spolu korelují atributy realLoans, x3059, x6089, x90.

Korelaci vytvořím ve statistickém systému R (viz zdrojový kód 3). Na řádku 5 načtu data uložená v CSV souboru. Data v záhlaví obsahují názvy sloupců a hodnoty jsou odděleny čárkou. Následovně data uložím do datové struktury `data.frame`, podobající se databázové tabulce nebo matici. Na řádku 8 a 9 specifikuji, které sloupce si přeji vybrat a následně nad nimi provedu výpočet korelace. Nakonec výstup uložím do souboru.

```
1  correlation <- function() {
2    inputPath <- "/home/username/output"
3    outputPath <- "/home/username/input"
4
5    csvData <- read.csv(file = inputPath, head = TRUE, sep = ",")
6    dataFrame <- data.frame(csvData)
7
8    columns <- c("age", "income", "pastDeliquency", "wholeDeliquency", "
         debtRatio", "openLoans", "realLoans", "dependents", "x3059", "
         x6089", "x90")
9    dataFrame <- dataFrame[columns]
10
11   corelation <- cor(dataFrame)
12
13   sink(outputPath)
14         cat(as.table(corelation))
15   sink()
16 }
17
18 correlation()
```

Zdrojový kód č. 3: Korelace proměnných.

4.2 Skrytá hrozba

Může vzniknout dojem, že pokud znám dominantní atributy, které klasifikátor ovlivňují, mohu rovnou přejít k modelování. Nabízí se mi možnost provést segmentaci podle korelujících atributů v hodnotách, ve kterých dominují. Například si vybrat klienty, kteří mají atribut realLoans roven nule a atributy x3059, x6089 a x90 jsou větší nebo rovny nule. Dále vybrat takové klienty, kteří mají atribut realLoans

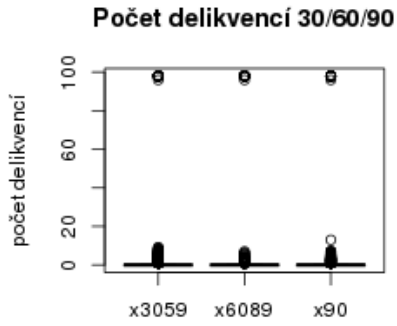
roven jedné, mají alespoň jeden nebo více delikvencí 30, ale žádnou nesplacenou pohledávku v 60 nebo 90 dnech po splatnosti. Tyto segmentace předložím v IBM SPSS Modeler rozhodovacím stromům, provedu vygenerování modelů a reportů jak klasifikace proběhla. Skrytá hrozba spočívá v tom, že takto klasifikátor nebude správně fungovat. Důvod, proč nebude klasifikátor správně fugovat, je nepoměr mezi atributem pastDeliquency, který nabývá hodnotu nula a jedna. Atribut pastDeliquency v hodnotě nula představuje přibližně 7,5% všech pozorování a v hodnotě nula tvoří 92,5%, což je značný nepoměr. V takovém případě se klasifikátor nemůže správně „naučit“ klasifikovat, jelikož všechny záznamy klasifikuje jako nula.

4.3 Tvorba vlastní trénovací množniny

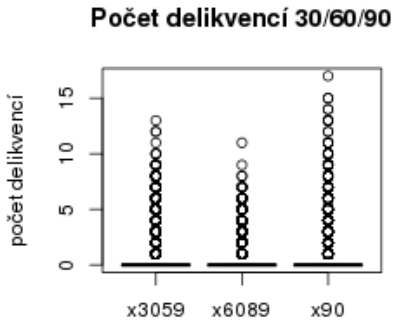
Vzhledem k hrozbě popsané v kapitole 4.2 si vytvořím vlastní trénovací množinu. Strategií, jak tuto množinu vytvořit se nabízí několik.

1. Jako základ vyberu všechny klienty, kteří mají pastDeliquency roven jedné a náhodně vyberu stejný počet klientů, kteří mají atribut pastDeliquency roven nule. Nevýhoda tohoto řešení je, že prakticky 99% klientů má hlavní korelující atributy rovny nule. Je třeba mít zastoupeny i ty klienty, kteří toto nesplňují.
2. Základem budou klienti, kteří mají pastDeliquency roven jedné a postupně k nim budu vybírat stejný počet klientů, jejichž pastDeliquency je roven nule. Takovýchto trénovacích množin vytvořím třináct. První nevýhoda tohoto přístupu je rozhodnutí, která z těchto třinácti množin je nejideálnější. Druhý problém je totožný s nevýhodou uvedenou v bodě č. 1.
3. Jako základ budu opět uvažovat všechny klienty, jejichž atribut pastDeliquency je roven jedné a k nim si uměle vygeneruji stejný počet klientů, jejichž pastDeliquency bude roven nule. Toto řešení není možné akceptovat, jelikož náhodným generováním se poruší asociace mezi všemi atributy.
4. Základem opět budiž klienti, kteří mají atribut pastDeliquency roven jedné. Pro doplnění trénovací množiny bude každému ze čtyř atributů poskytnut prostor 25% pro reprezentování celé šířky svého spektra.
5. Myšlenka čerpá z bodu 4., kde do vytvářené trénovací množiny zahrnuji všechny atributy i odlehlá pozorování. Odlehlých pozorování je přibližně 1%. I kdybych předpokládal, že klient, který mnohokrát nesplatil svoje závazky po třiceti dnech od spaltnosti je špatný klient, rozhodovací strom by se na tak malém počtu pozorování „nenaučí“ rozeznávat dobré a špatné klienty.

Trénovací množinu vytvořím na základě bodu 5. Na obrázku 5 je zřetelné, že delikvence v původní trénovací množině u některých klientů dosahuje až sto výskytů neplnění závazků po třiceti, šedesáti nebo devadesáti dnech po splatnosti. Naopak na obrázku 6 jsou klienti, u kterých se plnění a neplnění závazků více blíží zkutečnosti.



Obrázek č. 5: Původní trénovací množina.



Obrázek č. 6: Vlastní trénovací množina.

Boxplot nebo-li krabicový graf je grafická vizualizace ukazatelů polohy. Boxplot je zdola ohraničen 1. kvantilem, zhora potom 3. kvantilem. Mezi těmito kvantily se nachází čára, která zobrazuje medián. Mimo boxplot se zobrazují minimální a maximální hodnoty souboru dat a odlehlá pozorování.

Boxplot se ve statistickém systému R vytváří pomocí funkce `boxplot()`, která se ve zdrojovém kódu 4 nachází na řádku 11. Jako první parametr příjmá datovou strukturu `data.frame`, ve které jsou uložené sloupce x3059, x6089 a x90. Druhý parametr je vektor názvů, jak mají být boxploty na ose x popsány. Posledním parametrem je popis osy y. Funkce `boxplot()` je obalena funkcí `png()` a `dev.off()`, které obrázek uloží na disk.

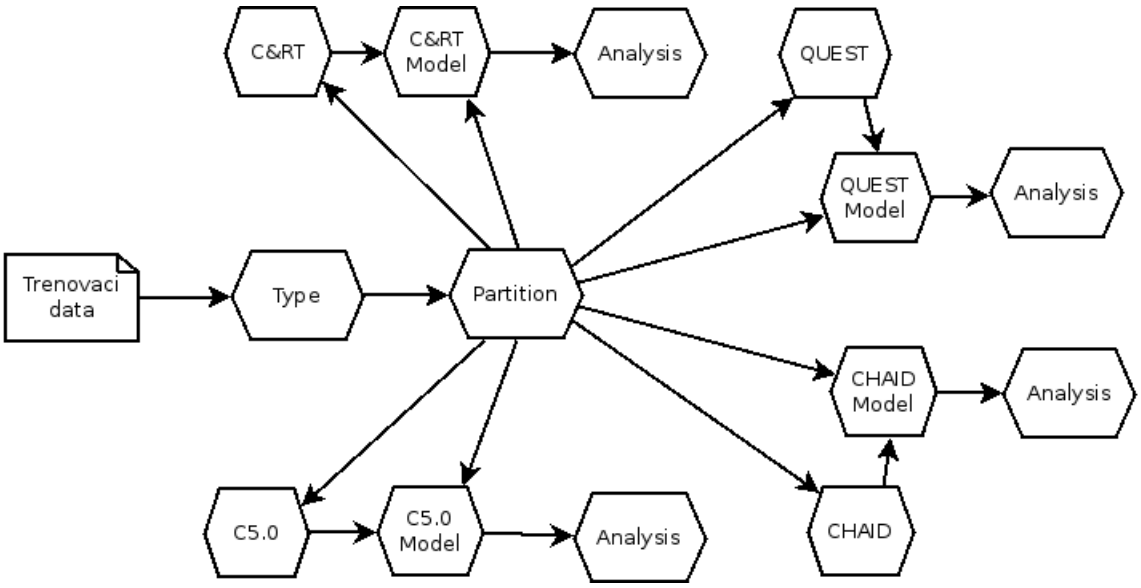
```
1  boxPlot <- function() {
2          originalTrainSet <- "/home/username/data"
3
4          csvData <- read.csv(file = originalTrainSet)
5          dataFrame <- data.frame(csvData)
6
7          x306090 <- dataFrame[,c("x3059", "x6089", "x90")]
8          namesOf306090 <- c("x3059", "x6089", "x90")
9
10         png(filename = "/home/username/originalTrainSet.png", width =
               250, height = 250)
11                 boxplot(x = x306090, names = namesOf306090, ylab = "
                       pocet delikvenci")
12                 title("Pocet delikvenci 30/60/90")
13         dev.off()
14 }
15
16 boxPlot()
```

Zdrojový kód č. 4: Princip tvorby boxplotů v R.

4.4 Trénování a výsledky modelování

Trénování modelů rozhodovacích stromů na trénovacích datech provedu v IBM SPSS Modeleru. IBM SPSS Modeler implementuje tyto čtyři rozhodovací stromy: C&RT, QUEST, CHAID a C5.0.

Analytický software IBM SPSS Modeler dovoluje vytvořit vizuální schéma, které označuje jako *stream*. Do streamu se vkládají prvky, které tvoří proces dolování dat. Těto prvkům se říká *element* nebo *component*. Data vstupující do procesu se nacházejí v elementu Training data. Tento element se v IBM SPSS Modeler jmenuje `Variable File Node`. Odtud data přechází do komponenty `Type`, kde se specifikují jakého typu atributy jsou. V elementu `Type` nastavím atribut wholeDeliquency jako cílovou proměnnou. Ostatní atributy budou prediktivní. Následně data přecházejí do elementu `Partition`, který data rozděluje na testovací data, trénovací a validační data. Z elementu `Partition` data vstupují postupně do všech rozhodovacích stromů, které IBM SPSS Modeler nabízí. Stromy v této fázi modelování jsou ve výchozím nastavení. Strom data načte a podle svých parametrů vygeneruje model.



Obrázek č. 7: Stream trénování rozhodovacích stromů.

Typ stromu	**Trénovací množina [%]**	**Testovací množina [%]**	**Validační množina [%]**
C&RT	78,57	76,53	77,1
QUEST	75,45	75,36	75,15
CHAID	79,98	77,78	77,94
C5.0	**80,9**	**77,66**	**78,19**

Tabulka č. 4: Správná klasifikace.

Typ stromu	Trénovací množina [%]	Testovací množina [%]	Validační množina [%]
C&RT	21,43	23,47	22,9
QUEST	24,55	24,64	24,85
CHAID	20,02	22,22	22,06
C5.0	**19,1**	**22,34**	**21,81**

Tabulka č. 5: Chybná klasifikace.

4.5 Vylepšování rozhodovacího stromu C5.0

Nejlepších výsledů podle tabulky 4 a 5 dosáhl rozhodovací strom C5.0. Existují možnosti různě nastavovat parametry rozhodovacího stromu a tím dosáhnout lepších výsledků. Tabulka 6 shrnuje, jak jsem nastavil parametry rozhodovacího stromu v IBM SPSS Modeler. Tabulka 7 zobrazuje výslekdy klasifikace po nastavení příslušných parametrů.

- **Use partitioned data** je vhodné vybrat v případě, že ve streamu neexistuje element Partitioner, nebo-li rozdělení dat na testovací, trénovací a případně validační podmnožinu.
- **Build model for each split** vyjadřuje možnost vytvořit model pro každý atribut, u kterého dochází k rozštěpení uzlu stromu. Potom se vybere ten nejlepší podstrom.
- **Group symbolics** je volba, která pokud je vybraná, strom se snaží kombinovat hodnoty, které mají stejné nebo podobné vlastnosti.
- **Use boosting** je speciální volba pro rozhodovací strom C5.0, která vylepšuje jeho přesnost. Princip je takový, že strom postupně vygeneruje N modelů. Počet modelů se nastavuje v *Number of trials*.
- **Number of trials** znamená, že první model je vytvořený tradiční cestou. Druhý model se zaměřuje na ty instance, které byly chybně klasifikované v prvním modelu. Třetí model se opět zaměřuje na chyby druhého modelu. Takto se iteruje přes všechny N modely.
- **Pruning severity** nebo-li prořezávání stromu. Hodnota se nastavuje v číselném intervalu 0 až 100. Pokud se hodnota zvětšuje, výsledkem je malý kompaktní strom, kde ale čistota listů není taková, jako v neprořezaném stromu. Snižováním hodnoty se strom více rozrůstá, komplikuje se, ale zvětšuje se čistota listů.
- **Minimum records per child branch** vyjadřuje počet instancí v uzlu stromu. Snižováním této hodnoty dochází k tvorbě uzlů, ve kterých je šum, odlehé nebo chybné pozorování. Naopak zvyšováním této hodnoty jsou uzly více konzistentní a zvětšuje se tendence mít v uzlu jednu třídu.

- **Use global pruning** je speciální volba prořezávání, která se provádí ve dvou fázích. První fáze je lokální prořezávání, ve které se prohledávají podstromy. Pokud snižují přesnost modelu, jsou vyřezány. Druhá globální fáze uvažuje strom jako celek. Podstromy, které nevnášejí do modelu přesnost jsou prořezány. Globální prořezávání se provádí automaticky. Pokud si uživatel přeje lokální prořezávání, je třeba tuto možnost vybrat.
- **Winnow attributes** je možnost prozkoumat užitečnost atributů vstupujících do rozhodovacího stromu. Atributy, které nehrají velký význam v tvorbě modelu jsou vyloučeny. Tato volba může být nápomocná, pokud do stromu vstupuje mnoho atributů. Zároveň zvolení volby snižuje přeučení modelu.

Volba	**Nastavení**
Use partitioned data	nezvoleno
Build model for each split	zvoleno
Group symbolics	zvoleno
Use boosting	zvoleno
Number of trials	30
Cross-validate	zvoleno
Number of folds	10
Mode	expert
Pruning serverity	75
Minimum records per child branch	2
Use global pruning	nezvoleno
Winnow attributes	zvoleno

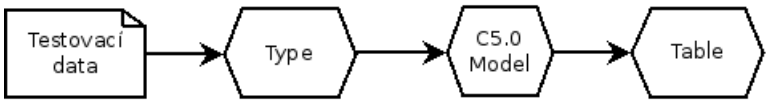
Tabulka č. 6: Zvolené nastavení rozhodovacího stromu C5.0.

Klasifikace	**Trénovací množina** [%]	**Testovací množina** [%]	**Validační množina** [%]
Správně klasifikované	81,94	81,59	82,06
Špatně klasifikované	18,06	18,41	17,94

Tabulka č. 7: Výsledky klasifikace po optimálním nastavení parametrů.

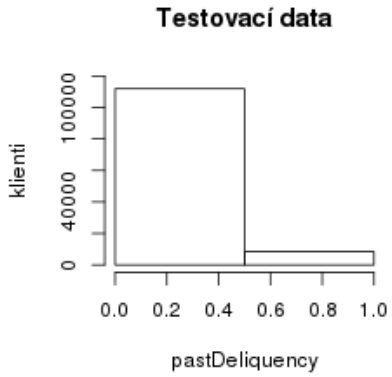
4.6 Aplikace modelu na testovací data

V IBM SPSS Modeler jsem vytvořil prakticky stejný stream jako v případě trénování modelů. Do Variable File Node vkládám testovací data. Data projdou elementem Type a následně vstoupí do mnou upraveného stromu C5.0 (viz kapitola 4.5). Model rozhodovacího stromu je napojen na element *Table*, který vrací data ve stejné podobě, jak do modelu vstupovaly a obohacuje je o nový sloupec. Je to sloupec, který představuje klasifikovanou cílovou proměnnou wholeDeliquency.

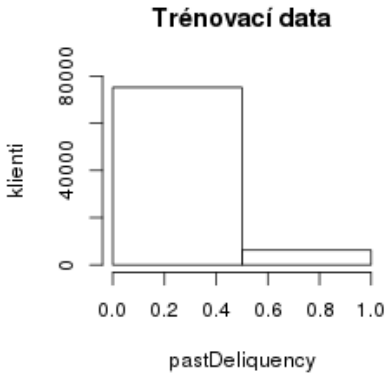


Obrázek č. 8: Stream trénování nad rozhodovacím stromem C5.0.

4.7 Výsledky testovacích dat na naučeném modelu



Obrázek č. 9: Histogram cílové proměnné v testovacích datech.



Obrázek č. 10: Histogram cílové proměnné v trénovacích datech.

Úkolem bylo klasifikovat proměnnou pastDeliquency. Hodnoty, kterých tento atribut nabývá vyjadřuje histogram na obrázku 9. Trénovací data, která IBM SPSS Modeler vydal na výstupu modelování jsem předložil statistickému systému R a vygeneroval jsem histogram pro trénovací data. Přesný počet klientů klasifikovaných jako „dobří klienti“ a „špatní klienti“ vyjadřuje tabulka 8. Špatné klienty chápeme jako ty, kteří nesplnili svoje závazky vůči finanční instituci po 90 dnech od splatnosti. Naopak dobří klienti jsou takoví, kteří se delikvence nedopouštějí.

	trénovací data		**testovací data**	
typ klientů	špatní klienti	dobří klienti	špatní klienti	dobří klienti
počet klientů	111912	8357	75166	6234

Tabulka č. 8: Přesné počty klasifikovaných klientů v cílové proměnné.

5 Závěr a návrhy na další práci

5.1 Závěr

Záměrem práce bylo porozumět datům týkající se fiktivních klientů finanční společnosti. Pohledy na tuto problematiku byly dva a to jak obecně z pohledu metod dolování dat, tak z pohledu credit scoringu.

Následně jsem syrová data připravil do vhodné podoby pro modelování. Za tímto účelem jsem naprogramoval vlastní nástroj, který data filtruje na základě zvoleného atributu a jeho hodnoty, zpracovává chybějící pozorování nebo vytváří nové odvozené atributy.

Před samotným modelováním jsem připravená a vyčištěná data analyzoval statistickým systémem R. Díky těmto analýzám jsem získal větší představu, jaká data modeluji a s jakými problémy se potýkám.

Vytvořil jsem vlastní trénovací množinu, kterou jsem v nástroji IBM SPSS Modeler předkládal rozhodovacím stromům C&RT, QUEST, CHAID a C5.0. Nejlepších výsledků dosáhl rozhodovací strom C5.0, u kterého jsem nastavoval parametry fungování, čímž jsem dosáhl lepších výsledků správně a chybně klasifikovaných klientů, kteří budou nebo nebudou plnit svoje závazky.

5.2 Návrhy na další práci

Na jedné straně by program DataFarmer mohl komunikovat s databázemi skrze moderní ORM[1] frameworky. Například Hibernate nebo MyBatis. Program by také mohl podporovat webové služby nebo práci s více druhy typů souborů. Například JSON nebo XML. Na druhé straně by program mohl skrze nějaké rozhraní nebo nativně podporovat práci se systémem R. Zajímavé by bylo rozvinout program DataFarmer jak po stránce textové, tedy všechnu funkcionalitu obalit nějakým příjemným textovým sekvenčním procesem zpracování dat, tak implementovat grafické uživatelské rozhraní.

Program IBM SPSS Modeler poskytuje vlastní programové rozhraní, skrze které by se dalo modelování automatizovat. Ve vyšších verzích tohoto software existuje i nativní podpora systému R nebo interpretovaného jazyka Python a to by mohla být cesta kvalitnější automatizace a integrace s okolím.

[1] Object-relational mapping

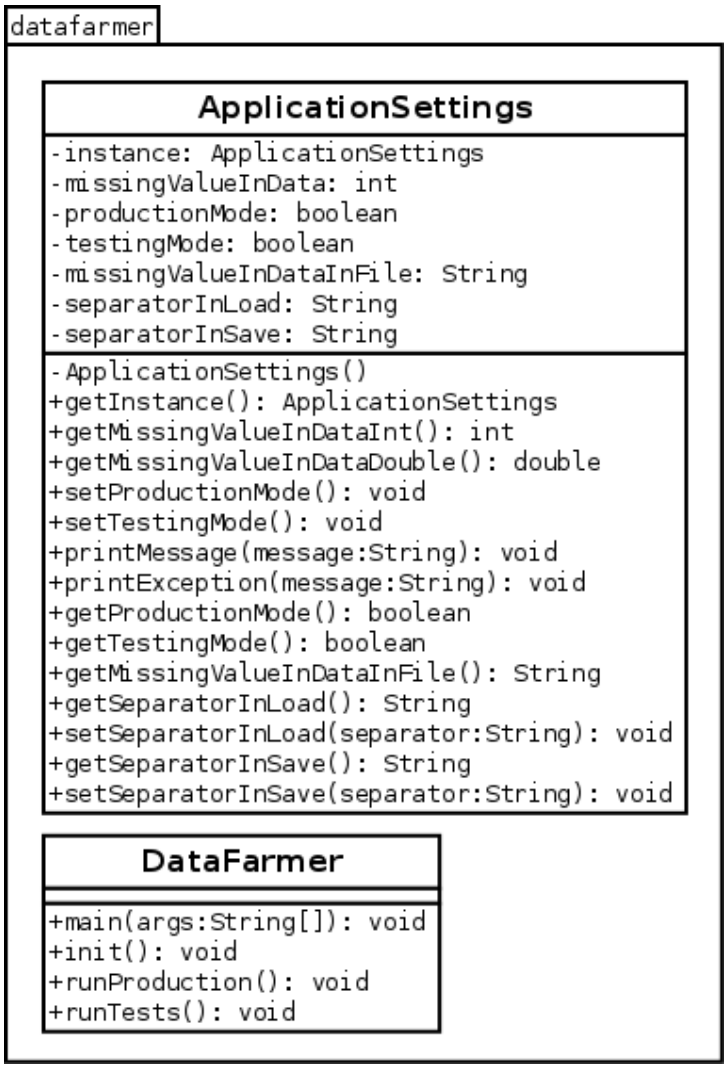
6 Seznamy

6.1 Seznam obrázků

6.2 Seznam tabulek

6.3 Seznam zdrojových kódů

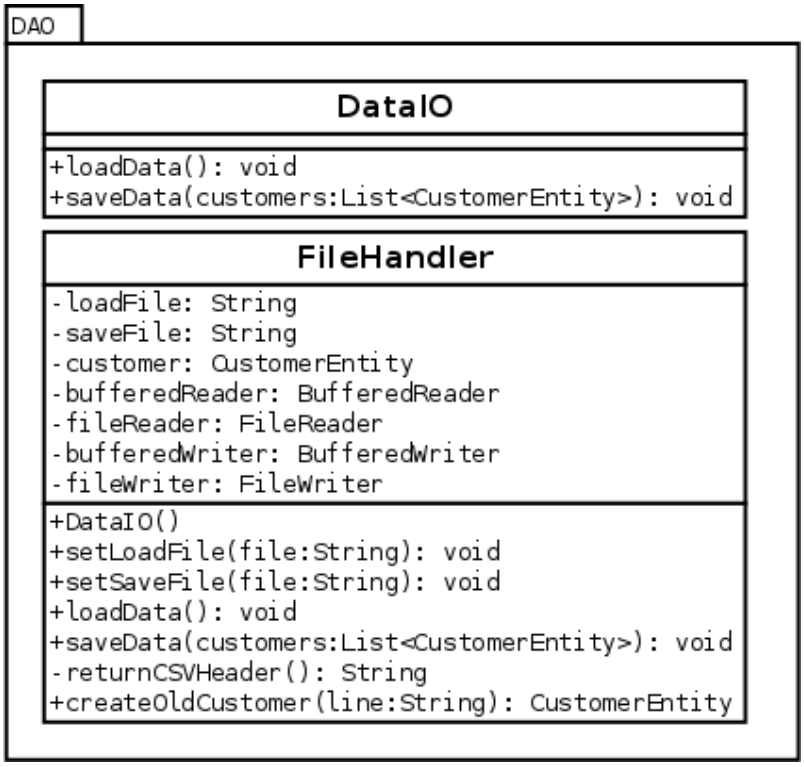
7 Přílohy



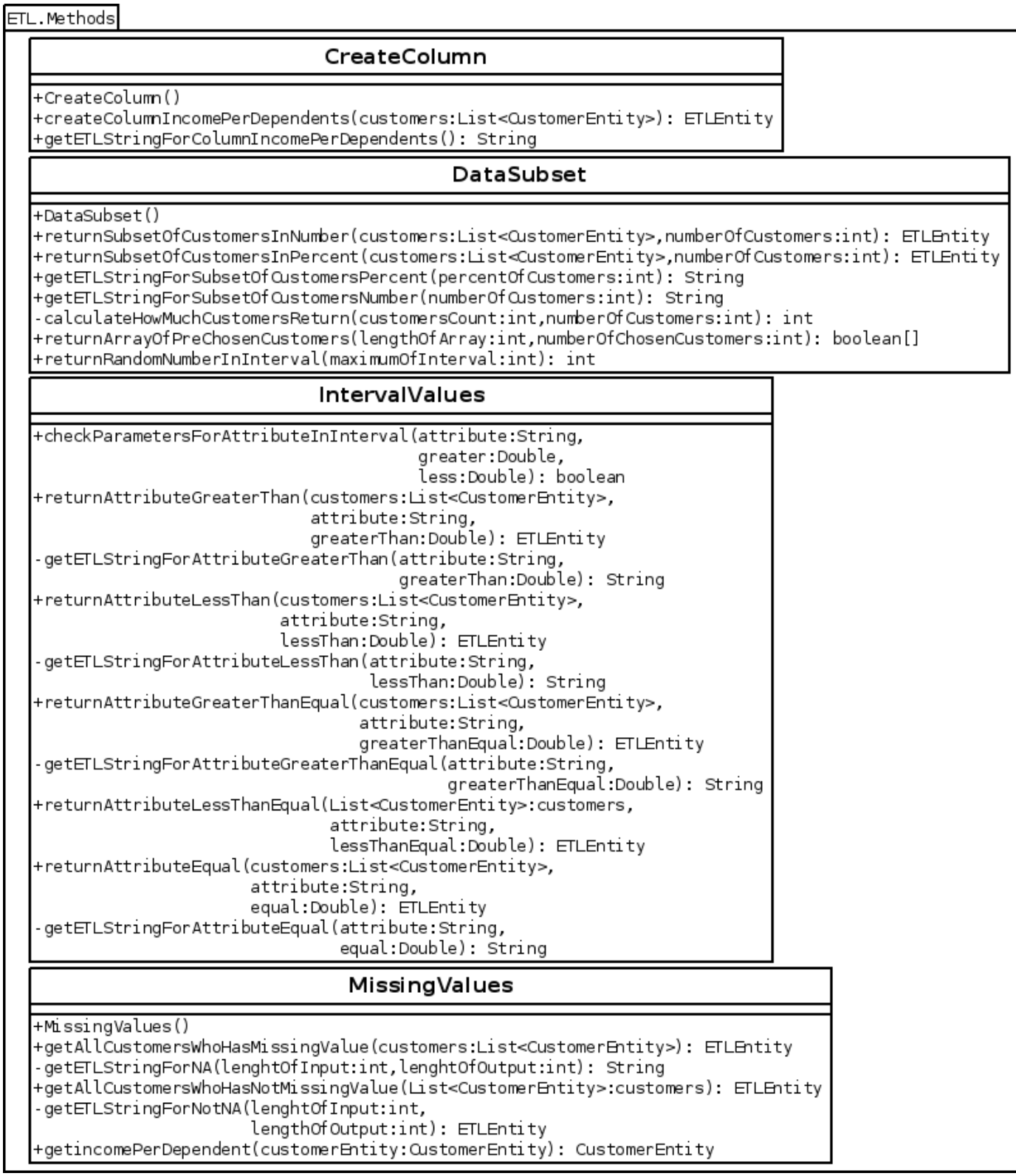
Obrázek č. 11: Diagram tříd ApplicationSettings a DataFarmer.

Obrázek č. 12: Diagram tříd CustomerEntity a CustomerRawData.

Obrázek č. 13: Diagram tříd ETL a ETLEntity.



Obrázek č. 14: Diagram tříd DataIO a FileHandler.



Obrázek č. 15: Diagram tříd CreateColumn, DataSubset, InternalValues a

MissingValues.

	age	income	pastDeliquency	wholeDeliquency	debtRatio	openLoans	realLoans	dependents	x3059	x6089	x90
age	1.000	0.038	-0.123	-0.013	0.000	0.189	0.065	-0.206	-0.048	-0.041	-0.047
income	0.038	1.000	-0.017	0.010	-0.052	0.112	0.170	0.091	-0.013	-0.014	-0.016
pastDeliquency	-0.123	-0.017	1.000	0.000	-0.003	-0.034	-0.002	0.044	0.158	0.130	0.146
wholeDeliquency	-0.013	0.010	0.000	1.000	-0.002	-0.015	0.004	0.013	-0.001	-0.001	0.001
debtRatio	0.000	-0.052	-0.003	-0.002	1.000	0.022	0.041	0.002	-0.005	-0.004	-0.005
openLoans	0.189	0.112	-0.034	-0.015	0.022	1.000	**0.417**	0.054	-0.049	-0.066	-0.077
realLoans	0.065	0.170	-0.002	0.004	0.041	**0.417**	1.000	0.122	-0.026	-0.037	-0.045
dependents	-0.206	0.091	0.044	0.013	0.002	0.054	0.122	1.000	0.006	0.000	0.000
x3059	-0.048	-0.013	0.158	-0.001	-0.005	-0.049	-0.026	0.006	1.000	**0.982**	**0.978**
x6089	-0.041	-0.014	0.130	-0.001	-0.004	-0.066	-0.037	0.000	**0.982**	1.000	**0.991**
x90	-0.047	-0.016	0.146	0.001	-0.005	-0.077	-0.045	0.000	**0.978**	**0.991**	1.000

Tabulka č. 9: Tabulka korelace atributů.

8 Reference

Michael J.A. Berry, Gordon S. Linoff *Data Mining Techniques For Marketing, Sales, and Customer Relationship Management* 2004. Wiley Publishing ISBN: 0-471-47064-3 .

Kim Hamilton, Russell Miles *Learnin UML 2.0*, 2006 O'Reilly Media, Inc. ISBN: 0-596-00982-8 .

Jiawei Han, Micheline Kamber, Jian Pei *Data Mining Concepts and Techniques* 2012. Morgan Kaufmann Publisher ISBN: 978-0-12-381479-1 .

Daniel T. Larose *Discovering Knowledge in Data*, 2005 Wiley Publishing ISNB: 0-471-66657-2 .

Roderic J. A. Little, Donald B. Rubin *Statistical Analysis With Missing Data*, 2002 Wiley Publishing ISNB: 0-471-18386-5 .

Rudolf Pecinovský *Návrhové vzory*, 2007 Computer Press ISBN: 978-80-251-1582-4 .

S. Sumathi, S. N. Sivanadam *Introduction to Data Mining and its Application*, 2006 Springer ISNB: 978-3-540-34350-9 .

Alexander J. Smola, S. V. N. Vishwanathan *Introduction to Machine Learning*, 2008 Cambridge University Press ISBN: 0-521-82583-0 .

Naeem Siddiqi *Credit Risk Scorecards, Developing and Implementing Intelligent Credit Scoring*, 2006 Wiley Publishing ISBN: 978-0-471-75451-0 .

Ian H. Witten, Eibe Frank, Mark A. Hall *Data Mining Practical Machine Learning Tools and Techniques* 2011. Morgan Kaufmann Publisher ISBN: 978-0-12-374856-0 .