VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

ANALÝZA FIREMNÍCH DAT O VYTÍŽENOSTI SLUŽEB S VYUŽITÍM DOLOVÁNÍ DAT

ANALYSIS OF COMPANY DATA ABOUT SERVICE WORKLOAD WITH USE OF DATA MINING

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. DAVID FOJTÍK**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. VLADIMÍR BARTÍK, Ph.D.**
SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2018/2019

Zadání diplomové práce

Student: **Fojtík David, Bc.**
Program: Informační technologie Obor: Informační systémy
Název: **Analýza firemních dat o vytíženosti služeb s využitím dolování dat**
Analysis of Company Data about Service Workload with Use of Data Mining
Kategorie: Data mining

Zadání:

1. Seznamte se s problematikou získávání znalostí z databází.
2. Podrobně prostudujte firemní data o vytíženosti služeb souvisejících s produkty a navrhněte potenciálně zajímavé úlohy získávání znalostí.
3. Ověřte úspěšnost metod pro tyto úlohy v existujících nástrojích (např. Rapid Miner).
4. Po konzultaci s vedoucím zvolte jednu z úloh a navrhněte koncepci aplikace, v rámci níž bude implementována. Využijte k tomu vhodné modelovací techniky.
5. Navrženou aplikaci implementujte ve vhodném prostředí, ověřte její funkčnost a proveďte experimenty ověřující úspěšnost zvolené metody.
6. Zhodnoťte dosažené výsledky a diskutujte další možné pokračování v tomto projektu.

Literatura:

- Han, J., Kamber, M.: Data Mining - Concepts and Techniques, 2nd Edition. Morgan Kaufmann Publishers, 2006.

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 3, částečně bod 4.

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Bartík Vladimír, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 22. května 2019
Datum schválení: 23. října 2018

Abstrakt

Tato práce spadá do oblasti získávání znalostí z databází. Dolování dat probíhá na reálných datech firmy o vytíženosti jejich služeb souvisejících především s jejich produkty. Byly vybrány tři úlohy, které jsou pro firmu potenciálně užitečné a měly by jí pomoci v rozhodování. První úloha se týká získání asociačních pravidel z produktů. Druhá úloha experimentuje s různými klasifikačními metodami a ověřuje, které z nich dávají nejlepší výsledky pro predikci, zda si zákazník s určitými parametry koupí daný produkt. Poslední úloha spadá do oblasti shlukování, kde se hledají odlehlé hodnoty (zákazníci) z dat jejich požadavků. Ve výsledné aplikaci je implementováno dolování asociačních pravidel pomocí algoritmů FP Growth a Apriori. Také je v aplikaci implementována druhá úloha a je použit predikční model Naive Bayes.

Abstract

This work falls within the field of acquiring knowledge from databases. Data mining takes place on real business data about the use of their services mainly related to their products. Three tasks were selected that are potentially useful to the business and should help in making decisions. The first task involves obtaining association rules from products. The second task experiments with different classification methods and verifies which ones give the best results to predict various customer purchases of concrete product with certain parameters. The last task falls into the clustering area where remote values (customers) are searched within the data of their requests. Association rules mining using algorithms FP Growth and Apriori is implemented in the resulting application. Also, the second task is implemented in the application and the Naive Bayes prediction model is used.

Klíčová slova

Datamining, RapidMiner, získávání znalostí, analýza firemních dat, asociační pravidla, klasifikace a predikce, shlukování, Apriori, FP Growth, Naive Bayes

Keywords

Datamining, RapidMiner, mining knowledge discovery, analysis of Company Data, association rules, classification and prediction, clustering, Apriori, FP Growth, Naive Bayes

Citace

FOJTÍK, David. *Analýza firemních dat o vytíženosti služeb s využitím dolování dat.* Brno, 2019. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vladimír Bartík, Ph.D.

Analýza firemních dat o vytíženosti služeb s využitím dolování dat

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Vladimíra Bartíka, Ph.D. Další informace a data mi byly poskytnuty od firmy, která vyvíjí mzdový a personální systém. Firma si však přeje být v práci neuvedena. Dále prohlašuji, že jména zákazníků nebyly nikde uvedeny, pracuje se v rámci experimentů pouze s jejich identifikačními čísly. Jména zákazníků je možné dohledat pouze interně v informačním systému firmy. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
David Fojtík
21. května 2019

Poděkování

Děkuji vedoucímu mé práce Ing. Vladimíru Bartíkovi, Ph.D. za odborné přípomínky, konzultace a rady během vedení této práce.

Obsah

Kapitola 1

Úvod

S postupem času a rozvojem technologií začal velice rychle vznikat směr, který se nazývá získávání znalostí z databází. Získávání znalostí z databází je extrakce zajímavých vzorů z velkého objemu dat, které nejsou na první pohled vidět. Získaná znalost pak může efektivně posloužit v procesu rozhodování. Mezi typické příklady patří rozhodnutí o poskytnutí úvěru klientovi, dále analýzy nákupního košíku nebo řízení rizik. Mohou to být ale i získané znalosti z oblasti bioinformatiky či odhalování podvodů.

Existuje řada alternativních jmen pro získávání znalostí z databází. Jde především o pojem *dolování dat*, který se v současné době používá častěji než získávání znalostí z databází.

Tato práce se zabývá analýzou firemních dat o vytíženosti služeb s využitím dolování dat. Zaměřil jsem se na produkty, které firma nabízí svým zákazníkům. Cílem této práce je pomoci firmě nalézt užitečné informace v jejich datech. Informace by firmě mohly posloužit jak při procesu rozhodování, tak i v oblasti vytváření zisku. Firma mi poskytla data svých produktů, zákazníků a data s požadavky od svých zákazníků. Neexistuje u nich zatím žádný nástroj, který by jim pomohl v rozhodování, který nový produkt by si stávající zákazník mohl zakoupit, jaké produkty se prodávají společně, případně jaká je pravděpodobnost, že si zákazník s určitými parametry koupí daný produkt. Na základě toho vznikly tři úlohy, které by měly být pro firmu přínosné.

Kapitola 2 pojednává o potřebné teorii z oblasti získávání znalostí z databází. Je zde popsán proces získávání znalostí spolu s postupy a metodami.

V kapitole 3 jsou popsána firemní data, se kterými se v průběhu experimentů pracovalo. Jsou zde také popsány jednotlivé úlohy. První úloha se zabývá asociačními pravidly a nabídkou souvisejících služeb. Druhá úloha částečně vychází z první. Jsou v ní použity jednotlivé klasifikační metody, které zjišťují, zda si zákazník s určitými parametry koupí daný produkt a s jakou pravděpodobností. Poslední úloha spadá do oblasti shlukování a pracuje s požadavky od zákazníků. Za pomocí odlehlých hodnot se snažím najít ty zákazníky, kteří se výrazně liší od ostatních.

Použité techonolgie a především nástroj Weka, který byl použit k implementaci výsledné aplikace, jsou popsány v kapitole 4. Implementace výsledné aplikace je popsána v kapitole 5. Předposlední kapitola 6 se zabývá ověřením funkčnosti. Závěrečná kapitola 7 shrnuje dosažené výsledky, užitečnost aplikace a možné pokračování v tomto projektu.

Kapitola 2

Získávání znalostí z databází a typy dolovacích úloh

Tato kapitola obsahuje popis potřebné teorie z oblasti získávání znalostí z databází. Nejprve je vysvětleno, co je samotné dolování, dále je popsán proces získávání znalostí a typy dolovacích úloh. Teorie, která je v této kapitole, vychází z velké části z knih [5],[17] a ze studijní opory do předmětu ZZN[1] [18].

2.1 Co je získávání znalostí

V posledních letech dochází k narůstání objemu dat a potřebě přeměnit tato data na užitečnou informaci a znalost. Získávání znalostí se dá uplatnit v širokém spektru různých odvětví. Například analýza trhu, detekce podvodů, zisk zákazníků, analýza produktů či vědecké bádání. Ve směru počítačových věd proto začal s postupem času velice rychle vznikat směr získávání znalostí z databází (obecně z dat) a s tím i související vznik nové generace podpůrných nástrojů pro analýzu dat.

Získávání znalostí z databází lze chápat jako výsledek přirozeného vývoje databázové technologie. Z prvopočátku se jednalo o podporu ukládání perzistentních dat, hierarchické a síťové databáze, relační databáze a odpovídající systémy řízení báze dat, přes rozvoj metod a nástrojů na podporu relačních databází, objektově orientovaný model dat nebo deduktivní, prostorové, temporální, vědecké databáze až po budování datových skladů.

Dolování je možné provádět prakticky na jakýchkoliv datech. Data z relačních databází informačích systémů nabízejí velké množství dat týkajících se například zákazníků, produktů či nákupů a prodejů, které lze pro dolování využít. Datové sklady umožňují shromažďovat a současně připravit data získaná z různých datových prostředí a zdrojů (zejména z produkčních databází) pro analýzu. Hlavním účelem produkčních databází je zajištění databázových transakcí. Tato činnost se označuje OLTP[2]. Z hlediska výkonu a zátěže je neúnosné provádět analýzy nad produkčními daty. Navíc často potřebujeme pro účely analýzy historická data, která nepotřebujeme v produkční databázi. Proto nám datové sklady slouží k oddělení těchto dvou prostředí.

Databáze časových řad obsahují posloupnosti hodnot získané například měřením, mohou to být také burzovní data nebo data o počasí. Model dat v datovém skladu má podobu vícerozměrné datové kostky, nad kterou jsou definovány určité operace, které slouží k mani-

[1] ZZN - Získávání znalostí z databází.
[2] OLTP - Online Transaction Processing.

pulaci s datovou kostkou. Taková oddělená analýza s odpovídajícími operacemi se označuje jako OLAP[3].

Transakční databáze obsahuje záznamy o transakcích, kde se každý záznam skládá z identifikátoru a seznamu položek. V tomto případě se většinou jedná o obchodní transakce, které udávají, jaké konkrétní produkty si zákazník v rámci transakce zakoupil.

Z textových a multimediálních databází může dolování probíhat podle obsahu nebo hledání podobností v obrazových datech.

Dolování je možné provádět z datových proudů, které obsahují data síťového provozu, video streamy nebo vědecká data.

Další velkou oblastí potenciálních aplikací získávání znalostí z databází jsou finanční analýzy a řízení rizik, detekce podvodů a neobvyklého chování či oblast bioinformatiky.

V poslední řadě se také uplatňuje dolování v prostředí webu. Provádí se analýza webových stránek nebo segmentace uživatelů sociálních sítí.

Výsledkem dolování je informace, která usnadní a pomůže v pozdějším rozhodování. Typickým příkladem je rozhodnutí o poskytnutí úvěru nějakému klientovi, analýzy nákupního košíku nebo v oblasti vědy může být výsledek dolování dat podkladem pro novou hypotézu, případně může potvrdit či vyvrátit již dříve stanovenou hypotézu.

Získávání znalostí z databází je extrakce (neboli "dolování") zajímavých dat a vzorů z velkých objemů dat, které jsou netriviální, skryté, dříve neznámé a pro nás potenciálně užitečné. Netriviální znalostí je v tomto kontextu myšlena informace, kterou nelze získat standardním přístupem, např. SQL dotazem nad databází, který zjistí počet prodaných produktů v určitém období nebo použití deduktivních databází či expertních systémů. Je nutné použít sofistikovaný postup. Modely a vzory jsou v datech skryté, to znamená, že nejsou na první pohled vidět, protože databáze k tomu nebyla navržena, a proto je musíme hledat.

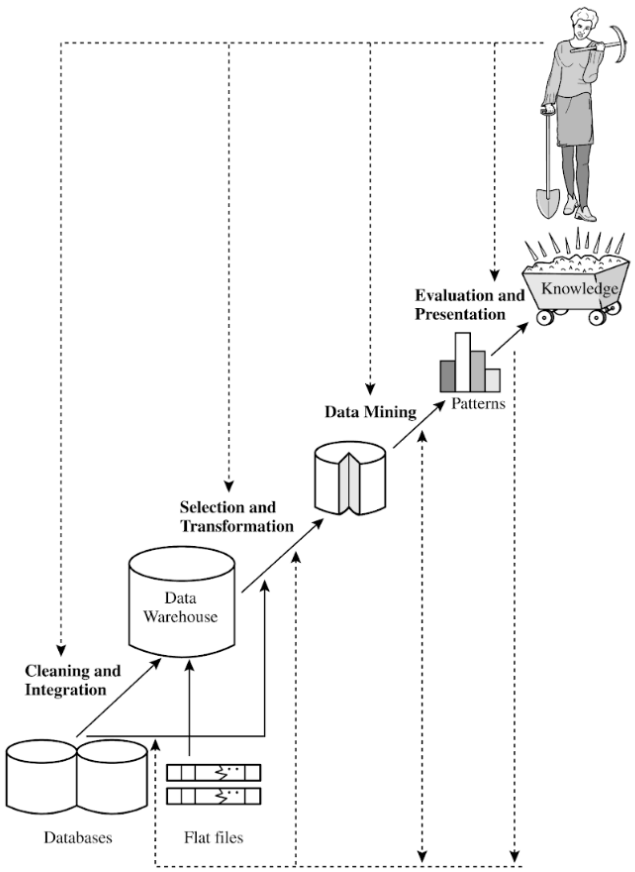
Jestliže chceme získat nějakou znalost z dat, neexistuje žádný univerzální postup, který by mohl být aplikovatelný na všechny problémy. Je nutné si určit cíl, který by měl být splněn, a na základě dat, které máme k dispozici, je nutné zvolit vhodnou dolovací techniku. Mezi základní a běžně používané techniky patří klasifikace a predikce, hledání frekventovaných vzorů (dolování asociačních pravidel), shluková analýza a s tím související detekce odlehlých objektů.

2.2 Proces získávání znalostí

Získávání znalostí z dat nezahrnuje jen samotné dolování (ve smyslu aplikace dolovacího algoritmu). Je to komplexní a náročný proces, který sestává z řady kroků. Nejprve probíhá příprava dat, která může být i nejnáročnější z celého procesu, dále samotné dolování a na konci vyhodnocení a prezentace výsledků. Jednotlivé kroky znázorňuje obrázek 2.1. Před samotným dolováním se provádí předzpracování dat. Jako předzpracování dat jsou označovány kroky zahrnující čištění, integraci, výběr a transformaci dat. Cílem předzpracování je zajistit co nejvyšší kvalitu dat, aby byly výsledky dolování co nejpřesnější a nebyly zkreslené. Všechny jednotlivé kroky se zpravidla v určitých iteracích opakují a nemusejí se nutně provádět v uvedeném pořadí.

Mezi kritéria charakterizující zajímavý vzor patří například jednoduchá srozumitelnost pro uživatele, novost či potencionální užitečnost.

[3]OLAP - Online Analytic Processing.



Obrázek 2.1: Proces získávání znalostí z databází (převzato z [5])

Čištění dat

- Jedná se o základní operaci, která bývá velice často vyžadována.
- Cílem čištění dat je vypořádat se s chybějícími položkami, odstranění šumu (položky navíc).
- V tomto kroku se také řeší nekonzistence dat a její odstranění.

Integrace dat

- Data většinou mohou pocházet z různých datových zdrojů. Cílem je tedy integrovat tato data do jednotného úložného prostoru – datového skladu.
- Integrace a čištění se často provádí společně, abychom docílili konzistence dat.

Výběr dat

- Cílem je vybrat ta data, která nás budou v rámci dané analytické úlohy zajímat a jsou pro nás relevantní.
- Z dat uložených v relačních databázích typicky získáváme znalosti z jedné tabulky a vybíráme sloupce, které nás zajímají.
- Pokud pracujeme s datovým skladem, můžeme analogicky vybírat konkrétní dimenze.

Transformace dat

- Transformace dat je proces, který umožňuje data transformovat do konsolidované podoby, která je vhodná pro dolovací metody. Běžnými operacemi jsou sumarizace nebo agregace.
- U datových skladů může transformace předcházet výběru dat, protože se může podílet právě na tvorbě datového skladu.

Dolování dat

- Je jádrem celého procesu získávání znalostí. Jedná se o aplikování konkrétní metody a algoritmu. Cílem je extrakce vzorů z dat, případně vytvoření modelu dat.

Hodnocení modelů a vzorů

- Cílem je identifikovat skutečně zajímavé vzory pomocí měr užitečnosti.

Prezentace znalostí

- Výsledky dolování jsou prezentovány uživateli. Využívají se známé techniky vizualizace a reprezentace znalostí.
- Získané výsledky se zobrazí pomocí názorných grafů, vizualizací, tabulek apod.

Čištění a integrace dat

Data, která chceme zpracovávat, se v rozsáhlých databázových systémech či datových skladech běžně mohou vyskytovat v nekompaktním či nekonzistentním stavu. Jestliže se data zaznamenávají například z nějakého zařízení, může u zařízení dojít k poruše či výpadku určité služby. To může způsobit nekonzistentní stav dat. Jedná se o náhodné chyby tzv. datový šum (příliš velké odchylky hodnot). Rovněž data mohou chybět (může být způsobeno i chybou člověka), případně pokud nebylo povinné vyplňovat konkrétní hodnotu (např. v nějakém formuláři).

Chybějící hodnoty v datech lze řešit několika způsoby. Záznamy můžeme ignorovat, tedy odstranit. Pokud například v daném řádku chybí více hodnot, je tato metoda určitě vhodná. Jinak by se odstraňování mělo používat jen minimálně. Další možností je ruční doplnění chybějící hodnoty. Pravděpodobně ten, kdo by hodnoty ručně doplňoval, bude mít jisté povědomí o datech, jakou mají strukturu a význam. Doplňování by tedy mělo být korektní. Ovšem vzhledem k obrovskému množství dat, je tento způsob velice časově náročný a používá se jen minimálně. Doplňování hodnot můžeme zautomatizovat za použití například předem definované globální konstanty, která bude nahrazovat chybějící hodnotu. Pokud by však chybělo hodně hodnot u konkrétního atributu, nechtěně bychom si tímto způsobem zanesli umělou chybu. Lepší způsob než nahrazovat konstantou je nahrazovat např. průměrnou hodnotou atributu ze všech záznamů třídy nebo použití nejpravděpodobnější hodnoty (nejvhodnější způsob).

Zašumělá data obsahují nesprávné hodnoty nebo data, která jsou mimo rozsah. Příčinou vzniku může být opět chyba při sběru nebo zadávání dat, ale bývá to i chyba neshody formátů např. při slučování dat z více zdrojů. Každý zdroj totiž může používat různé konvence pojmenovávání atributů.

Pro ošetření zašumělých dat se používájí následující metody:

Plnění (binding)

- Pracuje se setříděnými daty.
- Data jsou rozdělena do tzv. košů a jsou rozdělena tak, aby každý koš obsahoval zhruba stejný počet hodnot, případně pouze hodnoty z daného intervalu.
- Hodnoty v každém koši se vyhlazují. Vyhlazení může být např. průměrem, mediánem nebo pomocí hraničních hodnot koše.

Regrese

- Snaží se odhadnout spojitou hodnotu atributu.
- Jádrem je regresní křivka a vyhlazení probíhá podle regresní funkce.
- Tato regresní funkce může být buď lineární (aproximace mezi dvěma body) nebo vícenásobná lineární regrese (více než dva prvky).

Shlukování

- Používá se, pokud předem neznáme třídu, podle které chceme data rozdělit.
- Tato metoda je vhodná pro hledání odlehlých hodnot, protože objekty, které mají podobné vlastnosti, tvoří jednotlivé shluky.
- Snaží se data rozdělit do shluků tak, aby variabilita dat v rámci konkrétního shluku byla co nejnižší, zatímco rozdíly mezi jednotlivými shluky co nejvyšší.
- Zjistí data, která mají něco společného, aniž bychom předem věděli, do jakých skupin bychom měli vzorek dat rozdělit.
- Hodnoty, které leží mimo shluk, je pak možné považovat za odlehlé hodnoty.

Jelikož data mohou pocházet z různých zdrojů, v každém zdroji může být použit jiný formát uložení atributů (např. datum může být někde uloženo ve formátu `01.01.2019`, jinde zase jako `01. ledna 2019`), je potřeba zajistit správné spárování těchto atributů s rozdílným formátem. Jak již bylo řečeno, v každém zdroji mohou být zavedeny jiné konvence pojmenování atributů, které rovněž vedou k problémům. Při integraci je také častým problémem redundance dat. Ta se detekuje například pomocí korelační analýzy. Příčinou redundance mohou být atributy, které popisují stejnou skutečnost odlišným způsobem nebo atributy, jejichž hodnoty lze odvodit z ostatních atributů.

Výběr, transformace a redukce dat

Při dolování informací z dat se většinou využívá jejich určitá podmnožina, která se daného problému týká, nikoliv všechna data. Určení konkrétní podmnonžiny se provádí v rámci předzpracování dat. Data, která budou využita při dolování, se vyberou z databáze nebo datového skladu.

Transformace dat je proces, který zahrnuje operace pro převod dat do vhodného tvaru, aby bylo možné použití vhodné dolovací techniky. Operace, které tento proces zahrnuje, jsou vyhlazování, agregace, generalizace, normalizace a konstrukce atributů.

Vyhlazování již bylo zmíněno výše v sekci čištění dat. Jeho cílem je odstranění šumu z dat.

Agregace se uplatňuje u datových kostek, které umožňují pohled na data v různých úrovních podrobnosti. Jedná se o souhrnné operace, které seskupí data podle určitých vlastností. Příkladem může být seskupení dat podle času, například na týdenní, měsíční či roční pohled.

Generalizace umožňuje zobecnění dat na základě konceptuální hierarchie. Typickým příkladem je zobecnění měst na jednotlivé kraje případně země.

Operace normalizace převádí data do menších specifických intervalů. Většinou se jedná o intervaly (-1,1) případně (0,1).

Konstrukce atributů nebo také někdy nazývána konstrukce rysů, slouží k vytvoření nových atributů, které jsou získány z dosavadních hodnot. Cílem konstrukce je zvýšení přesnosti dolování informací z dat.

Redukce dat se provádí především kvůli zmenšení objemu dat určených pro dolování. Redukce však musí zachovat vlastnosti původního souboru tak, aby výsledky dolování nad redukovaným souborem nebyly rozdílné. Mezi techniky redukce patří například výběr jen určité podmnožiny dat, kdy se odstraňují nevýznamné nebo nesouvisející atributy.

Interpretace výsledků a jejich prezentace

Nedílnou součástí procesu získávání znalostí je i interpretace a prezentace výsledků. Hodnocení úspěšnosti se liší podle použitých metod.

Existují různé přístupy k určení přesnosti klasifikátorů. Nejjednodušší způsob měření je spočtení podílu správně předpovězených příkladů z neznámcých testovacích dat. Běžně se používá dataset, který je rozdělen na dvě části. Na data, která slouží pro natrénování modelu, a potom na data, která se použijí pro otestování modelu. U testovacích dat jsou totiž známy všechny hodnoty atributů a dle výsledků modelu je možné porovnávat, jak správně náš klasifikátor zařazuje. Většinou se provádí náhodný výběr vzorku dat pro trénovací sadu a pro testovací sadu.

Pro metodu regresní analýzy, který udává velikost chyby při predikci spojité hodnoty, se používá hodnota mean squared error [17].

Jednotlivé modely se také liší mírou čitelnosti pro uživatele. Například rozhodovací stromy jsou zpravila jednoduše prezentovatelné a čitelné, u neuronových sítí je vytvoření čitelných a jednoduchých pravidel téměr nemožné. Pro prezentaci výsledků je vhodné použít prvky vizualizace, jako je barevné odlišení významných hodnot nebo odchylek, případně data vynést do vhodného grafu nebo tabulky.

2.3 Typy dolovacích úloh

Je celá řada nejrůznějších metod a postupů, jak provádět dolování znalostí z dat. Existuje rovněž spousta aplikací, které nám dolování ulehčují. V této části jsou charakterizovány základní typy dolovacích úloh, které jsou řešeny v kroku dolování dat celého procesu. V podstatě jde o to, jaký druh modelu dat se snažíme z dat získat.

Typy dolovacích úloh můžeme rozdělit na dvě skupiny:

Deskriptivní

- Charakterizují obecné vlastnosti analyzovaných dat.
- Typickým příkladem je analýza nákupního košíku a zjištění produktů, které se kupují společně.

Prediktivní

- Na základě analýzy současných dat se provádí dedukce pro předpověď budoucího chování.
- Typickým příkladem těchto úloh je klasifikace, která nám například umožní na základě dosavadního chování zákazníků předpovědět, jak se bude chovat zákazník nový.

2.3.1 Popis konceptu/třídy

Jedním ze základních typů dolovacích úloh je popis konceptu/třídy. Data mohou být asociována s určitou třídou nebo konceptem, mohou existovat třídy produktů nebo být zavedeny nové pojmy, týkající se zákazníků. Potom může být užitečné popsat třídy a koncepty souhrnným, stručným, ale dostatečně vystihujícím způsobem. Takový způsob nazýváme popis třídy/konceptu.

Popis můžeme získat jedním ze dvou následujících způsobů:

Charakterizace dat

- Značí sumarizaci obecných vlastností analyzované třídy. Data odpovídající třídě se z databáze vybírají pomocí jednoduchého dotazu.
- Jako příklad můžeme uvést zvýšení prodeje konkrétního produktu v určitém období.

Diskriminace dat

- Data analyzované třídy nepopisujeme obecnými výrazy, ale vymezujeme vztah k jedné nebo několika rozdílovým třídám. Hledáme tedy atributy, jejichž hodnoty se co nejvíce liší.
- Jako příklad můžeme uvést, jak se liší produkty, u kterých vzrostl prodej, od jiných produktů, u kterých naopak prodej klesl v konkrétním časovém období.

2.3.2 Asociační pravidla

Dalším typem dolovacích úloh jsou úlohy zaměřené na dolování vztahů mezi atributy. Patří sem dolování frekventovaných vzorů, asociací a korelací. Frekventované vzory jsou vzory, které se v datech vyskytují často, tzn. vyskytují se v daném souboru dat s určitou frekvencí. Mezi typickou úlohu patří analýza nákupního košíku, kde se jedná o frekventované množiny. Frekventované množiny jsou seznamy produktů, které se vyskytují často, případně skupiny produktů, které se v košíku nacházely zároveň.

Dolování asociačních pravidel je technika dolování dat, která umožňuje odhalit zajímavé vztahy (asociace) v poměrně velké databázi. Nicméně zajímavost pravidla se odvíjí od toho, jaký problém chce uživatel řešit. Stávající přístupy používají různé parametry hledání zajímavých pravidel. Některé se zaměřují jen na jediný cíl (třídu) a kombinují dolování pravidel i s klasifikací. Hlavní předností dolování asociačních pravidel je to, že jsou vždy nalezena všechna zajímavá pravidla. Počet asociací však v rozsáhlejších databázích může být velmi velký. [10]

Předpokládejme, že chceme zjistit, které položky z nabízeného sortimentu se prodávají společně viz vzorec 2.1. Zjistíme, že v 1% ze všech analyzovaných dat si zákazníci kupují společně $produkt_1$ a $produkt_2$. Existují i nákupy zákazníků, kde se vyskytuje samostatně pouze $produkt_1$ nebo $produkt_2$. Dále zjistíme, že v 50% všech případů, kdy si zákazník koupí $produkt_1$, následně si také koupí $produkt_2$. Výsledkem těchto zjištění je takzvané asociační pravidlo.

Asociační pravidlo má potom následující tvar:

$$kupuje(X, produkt_1) \Rightarrow kupuje(X, produkt_2)[podpora = 1\%, spolehlivost = 50\%] \quad (2.1)$$

Toto asociační pravidlo obsahuje jeden predikát *kupuje*, jedná se tedy o jednodimenzionální asociační pravidlo. Většinou však pravidla obsahují více než jeden predikát, a proto jsou označována jako multidimenzionální asociační pravidla. Proměnná X zde značí zákazníka. Pravidlo nám říká, že zákazník, který si koupí $produkt_1$, má tendenci si zároveň s ním koupit $produkt_2$.

Parametry **podpory** (*support*) a **spolehlivosti** ($confidence$) nám určují, do jaké míry je významnost tohoto zastoupení. Podpora se udává jako relativní (v %) nebo absolutní (počet transakcí, nutno však znát celkový počet analyzovaných transakcí). Podpora vyjadřuje (v našem případě), že v 1% ze všech analyzovaných dat si zákazníci koupili $produkt_1$ a současně $produkt_2$.

Spolehlivost vyjadřuje významnost zastoupení položek na pravé straně pravidla v transakcích, kde se vyskytují položky z levé strany pravidla. Spolehlivost nám ohodnocuje stupeň jistoty platnosti daného asociačního pravidla a je definovaná jako podmíněná pravděpodobnost. Tedy v 50% nákupů, při nichž si zákazník koupil $produkt_1$, si hned také koupil $produkt_2$. Podpora a spolehlivost jsou příkladem měr, které používáme k poměřování zajímavých dolovaných asociačních pravidel.

Formální definice

- Nechť $I = \{i_1, i_2, i_3...\}$ je množina položek.
- Nechť D je množina transakcí, kde T je množina položek taková, že $A \subseteq I$.
- Každá transakce T má svůj unikátní identifikátor *TID*.
- Nechť A je množina položek. Říkáme, že transakce T obsahuje A, pokud $A \subseteq T$.
- Asociační pravidlo je implikace ve tvaru $A \Rightarrow B$, kde $A \subset T$, $B \subset T$ a $A \cap B = \varnothing$.
- Pravidlo $A \Rightarrow B$ má podporu v množině transakcí D, která je vyjádřena v procentech. Podpora tedy vyjadřuje, v jakém počtu transakcí v D je obsažena množina položek $A \cup B$.
- Pravidlo $A \Rightarrow B$ má v množině transakcí spolehlivost, která udává, kolik % transakcí, které obsahují A, obsahuje také B.

S využitím pravděpodobnosti lze tedy napsat:

$$podpora(X \Rightarrow Y) = P(X \cup Y) \quad (2.2)$$

$$spolehlivost(X \Rightarrow Y) = P(X|Y) \quad (2.3)$$

Další měrou asociačních pravidel je hodnota **Lift** (zdvih) a je definována následovně:

$$Lift(X \Rightarrow Y) = \frac{podpora(X \cup Y)}{podpora(Y) \ \times \ podpora(X)} \tag{2.4}$$

Je to poměr pozorované podpory k očekávané, jestliže X a Y byly nezávislé. Lift udává, jak moc jsou X a Y vůči sobě nezávislé. Rozsah se pohybuje v intervalu $(0, +\infty)$. Hodnota kolem $Lift = 1$ znamená, že X a Y jsou nezávislé, a proto pravidlo není moc zajímavé. [13]

Hodnota **Conviction** (přesvědčivost) je citlivá na směr pravidla, tj. $Conv(X \Rightarrow Y)$ není stejné jako $Conv(Y \Rightarrow X)$. Conviction vychází z definice implikace. Snaží se měřit míru implikace pravidla. [13] Conviction je definováno následovně:

$$Conv(X \Rightarrow Y) = \frac{(1 - podpora(Y))}{(1 - podpora(X \Rightarrow Y))} \tag{2.5}$$

Typy pravidel

Asociační pravidla můžeme rozdělit do kategorií následovně:

Podle typu hodnot v pravidle

- Pokud nás zajímá jen přítomnost nebo nepřítomnost položky v transakci, pak se asociační pravidlo nazývá booleovské.
- Pokud pravidlo vyjadřuje vztah mezi kvantitativními atributy (např. *cena*), pak se jedná o kvantitativní asociační pravidlo.

Podle počtu dimenzí

- Pravidlo $kupuje(produkt_1) \Rightarrow kupuje(produkt_2)$ je příklad jednodimenzionálního pravidla, protože obsahuje pouze jednu dimenzi *kupuje*.
- Pravidlo $kupuje(produkt_1) \wedge cena(300 - 1000) \Rightarrow kupuje(produkt_2)$ je příklad vícedimenzionálního pravidla, protože obsahuje jednu dimenzi *kupuje* a druhou dimenzi *cena*.

Podle úrovně abstrakce

- Dolování asociačních pravidel může probíhat na různých úrovních abstrakce.
- Výsledkem jsou pravidla reprezentující vztahy mezi položkami na různých úrovních.
- Příkladem různých úrovní abstrakce mohou být dimenze $kupuje(LCD\ TV)$ a $kupuje(TV)$.

Algoritmus Apriori

Algoritmus Apriori slouží pro získání frekventovancýh množin. Apriori pracuje na principu využití předchozí znalosti o frekventovaných množinách. Důležitou roli hraje tzv. apriori vlastnost, která udává, že všechny podmnožiny frekventované množiny musejí být také frekventované. Algoritmus probíhá v jednotlivých iteracích, kdy jsou získávány n-prvkové frekventované množiny. Skládá se ze dvou kroků, spojovacího a vylučovacího. Ve spojovacím kroku jsou generováni kandidáti na frekventované množiny. Ve vylučovacím kroku jsou odstraňovány množiny, které nejsou frekventované. Algoritmus předpokládá, že všechny položky v množině jsou lexikograficky seřazeny.

Princip algoritmu

- V první iteraci se prochází každá položka a spočte se její podpora.
- Poté se porovnají hodnoty podpory kandidátů s minimální podporou.
- Následuje generování kandidátů o velikosti o jedna větší než v předchozím kroku.
- Rovněž se vypočte podpora kandidátů průchodem databází a odstraní se nefrekventované množiny na základě minimální podpory.
- Konec algoritmu při nalezení kandidátů dané velikosti.

Generování asociačních pravidel z frekventovaných množin

- Generování se provádí s využitím rovnice pro výpočet spolehlivosti 2.6.
- Pro každou frekventovanou množinu l jsou generovány všechny její neprázdné podmnožiny.
- Pro každou podmnožinu s je vygenerováno pravidlo $s \Rightarrow (l - s)$ a podle rovnice je vypočtena spolehlivost.
- Pokud je spolehlivost vyšší než minimální, pak je pravidlo silné.
- Protože jsou pravidla generována z frekventovaných množin, je pro ně automaticky splněna podmínka minimální podpory.

$$conf(A \Rightarrow B) = P(B|A) = \frac{s(A \cup B)}{s(A)} \tag{2.6}$$

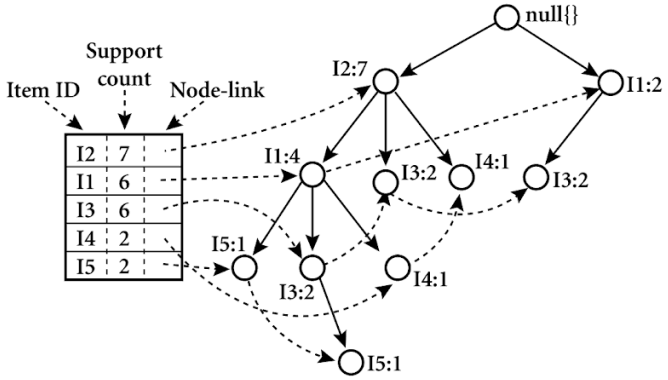
Nevýhodou algoritmu je právě vygenerování velkého množství kandidátů a také časté procházení databáze. Tyto nedostatky lze odstranit získáváním frekventovaných množin bez nutnosti generování kandidátů. K tomu slouží metoda vzrůstu frekventovaných množin, která využívá strukturu s názvem FP-strom.

FP-strom

FP-strom je datová struktura, která reprezentuje data ve stromové struktuře. Strom je tedy rozdělen na menší části tzv. podmíněné stromy, ze kterých jsou získávány frekventované množiny. Ty položky, které mají stejné podmnožiny, umožňují stromu zůstat kompaktním, protože jejich cesty se překrývají. Nejlepší případ, ke kterému může dojít je, když všechny transakce mají přesně stejnou sadu položek. Velikost FP-stromu bude pouze jedna

větev uzlů. Nejhorší scénář může nastat, když má každá transakce jinou sadu položek. Protože strom vyžaduje navíc uložení čítačů a ukazatelů na uzly, výsledný prostor pro uložení stromu je tím pádem větší, než prostor pro uložení původní datové sady. Složitost stromu roste s unikátností každé transakce. [6]

Na obrázku 2.2 je znázorněna reprezentace FP-stromu.



Obrázek 2.2: Reprezentace FP-stromu a tabulka odkazů na uzly (převzato z [5])

Konstrukce FP-stromu

- Vytvoří se kořen stromu označovaný jako *null*.
- Prochází se databáze a v každé transakci jsou zpracovávány položky dle počtu výskytů.
- Pro každou transakci je vložena větev, kde uzly jsou položky transakce.
- Jestliže některá transakce sdílí prefix dané větve, do nové větve se pouze přidá ta část, která není sdílená, a inkrementují se čítače u položek, které jsou stejné.
- U slučování větví tedy platí, že počet výskytů je inkrementován u všech položek, které tvoří společnou cestu grafem.
- Pro efektivnější průchod stromem je vytvořena tabulka odkazů na uzly a každý uzel obsahuje ukazatel na další uzel shodného jména.

Algoritmus FP-Growth

FP-Growth (z angl. frequent-pattern growth) transformuje problém hledání dlouhých frekventovaných vzorů na hledání vzorů kratších a připojování sufixů. Jako sufixy používá nejméně frekventované položky. Tato metoda značně snižuje časovou náročnost procesu dolování frekventovaných množin. Základem dolování je transformace databáze do výše zmíněné struktury FP-stromu.

2.3.3 Klasifikace a predikce

Mezi prediktivní úlohy dolování dat patří klasifikace a predikce. Cílem klasifikace je nalézt takový model dat, který popisuje a současně rozlišuje třídy dat. Model posléze používá k predikci třídy objektu, jehož zařazení neznáme.

Proces klasifikace sestává ze 3 kroků:

1. Trénování (učení) - na základě analýzy tzv. trénovací množiny je vytvořen klasifikační model.
2. Testování - hodnocení kvality vytvořeného modelu využitím testovacích dat.
3. Aplikace - použití natrénovaného modelu pro klasifikaci objektu, jehož třída není neznámá.

Většinou se použije datová sada, která je rozdělena poměrově na trénovací část a testovací část. U datové sady je známá třída, takže tuto informaci je možné využít pro natrénování a druhou pro otestování. Úkolem klasifikátoru je zjistit klasifikační pravidla, pomocí kterých se s jistou přesností může konkrétní objekt klasifikovat do dané třídy. Na základě výsledků testování, pokud vytvořený model neklasifikuje správně, je nutné upravit parametry nebo model vytvořit znovu. V kladném případě je model připraven k použití.

Kritériem, které by měly klasifikační a predikční metody splňovat, je přesnost předpovědi, tedy v kolika procentech daný model správně klasifikuje neznámá data. Další kritérium je výpočetní složitost výpočtu klasifikačních pravidel. Model by měl být robustní a měl by být vytvořen správně pro velké množství dat. Model by měl být pak dobře interpretován pro pochopení uživatelem.

Matice záměn

Matici záměn (*Confusion Matrix*) si můžeme představit jako kontingenční tabulku. Měření přesnosti klasifikátoru probíhá na datech, která nebyla použita pro učení, ale známe u nich přesnou třídu. Tabulka udává, v kolika případech se klasifikátor shoduje s učitelem a v kolika se dopouští chyby. Matice však může být libovolně velká.

Na následující ukázce je možné vidět matici o velikosti 2x2.

```
=== Confusion Matrix ===
a b <-- classified as
7 2 | a = yes
3 2 | b = no
```

Je zde třída *a* a třída *b*. Počet správně klasifikovaných instancí je na diagonálách v matici. Ostatní případy jsou špatně klasifikované instance. Hodnota *True Positive (TP)* je tedy počet příkladů, které byly zařazeny do třídy *a* a napříč všemi příklady mají opravdu správnou třídu. Naopak hodnota *False Positive (FP)* je počet příkladů, které byly klasifikovány jako třída *a*, ale místo toho patří do jiné třídy *b*. Třída *a* byla chybně klasifikována jako třída *b* ve dvou případech a naopak třída *b* byla chybně klasifikována jako třída *a* ve třech případech.

Techniky pro kombinaci více modelů

Během strojového učení je běžné, že se používají různé učící algoritmy, které se mezi sebou kombinují. Je to z důvodu zisku lepšího výkonu a hlavně přesnosti výsledků. Mezi nejpopulárnější techniky patří metody *Boosting*, *Bagging* a *Blending*. [2]

Boosting

- Boosting je metoda, ve které je použit nejprve jeden základní klasifikátor, pro který jsou připravena trénovací data. Druhý klasifikátor je pak vytvořen za ním a zaměřuje se konkrétně na ty instance z trénovacích dat, která první klasifikátor klasifikoval špatně.
- Proces přidávání klasifikátorů pokračuje tak dlouho, dokud nedojde k překročení limitu vytvořených modelů, případně dosažení požadované přesnosti.

Bagging

- Bagging je metoda, která vytváří oddělené vzorky trénovací datové sady a klasifikátor pro každý vzorek.
- Výsledky těchto vícenásobných klasifikátorů jsou pak kombinovány (např. zprůměrované).
- Díky tomu, že každý vzorek z trénovací datové sady je odlišný, dává každému dílčímu klasifikátoru možnost se zaměřit na daný problém z různých úhlů pohledu.

Blending

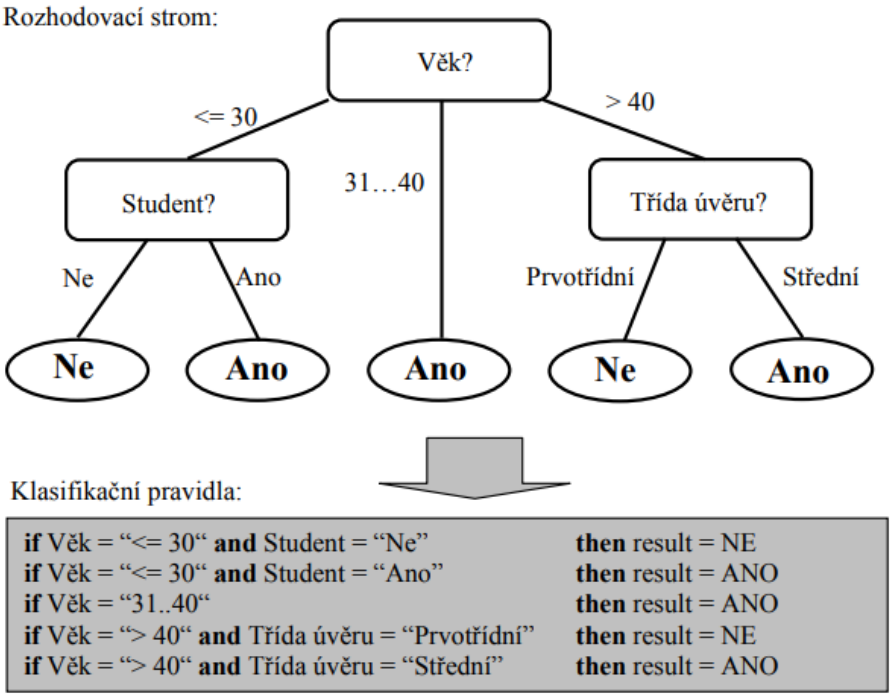
- Zahrnuje trénink klasifikátoru, který kombinuje predikce několika dalších klasifikátorů. Všechny ostatní klasifikátory jsou trénovány na stejných datech a hlavní klasifikátor se učí, jak zacházet s výsledky jednotlivých klasifikátorů, aby bylo dosaženo co největší přesnosti na neznámých datech.

Pokud máme omezený vstupní soubor s daty pro učení a chceme hodnotit úspěšnost klasifikace, provádíme **křížovou validaci**. Křížová validace je podrobnější metoda, kdy je vstupní dataset náhodně rozdělen a přeskupen na n částí (folds) stejné velikosti. V každé iteraci se každá jedna část použije pro testování a zbytek (n-1 částí) se použije pro trénování klasifikátoru. Výsledky testů v každé iteraci se shromažďují a následně se zprůměrují. Výsledkem je odhad přesnosti křížové validace. Jednotlivé části jsou buď čistě náhodné, případně mohou být mírně upraveny tak, aby vytvořily rovnoměrné třídní rozdělení z daného datasetu. [1]

Klasifikace pomocí rozhodovacích stromů

Klasifikaci je možné provádět pomocí rozhodovacích stromů. Jedná se opět o stromovou strukturu, ve které se nacházejí *nelistové uzly* obsahující jednotlivé testy na atributy, *listové uzly* uchovávající různé třídy, do kterých je objekt klasifikován. Takovýto strom pak může být jednoduše převeden na odpovídající klasifikační pravidlo.

Příklad rozhodovacího stromu je možné vidět na následujícím obrázku 2.3. Jsou zde klasifikováni zákazníci, kteří si budou či nebudou chtít koupit konkrétní produkt. Dále jsou zde vidět i odpovídající klasifikační pravidla vytvořena právě z tohoto stromu.



Obrázek 2.3: Rozhodovací strom s klasifikačními pravidly (převzato z [18])

Při vytváření rozhodovacího stromu mají opět velký vliv data, která jsou zašumělá. Mají vliv na větve, které dělají strom zbytečně složitější a navíc snižují přesnost predikce.

Základní metody eliminace těchto větví jsou následující:

Preprunig

- Nepotřebné větve jsou odstraňovány již při konstrukci stromu.
- Místo testu atributu na další podmínku, se tato část stromu ukončí listovým uzlem.
- Tento listový uzel se ohodnotí takovou klasifikační třídou, která je přiřazena nejvíce vzorkům odpovídající této části stromu.

Postpruning

- Rozhodovací strom je vytvořen celý.
- Posléze se nepotřebné větve odstraňují.
- Metoda je výpočetně náročnější, protože musí dojít k vygenerování celého stromu, avšak je spolehlivější.

Běžně se provádí kompromis a používají se obě metody zároveň.

Bayesovská klasifikace

Bayesovská klasifikace je klasifikační metoda, která je založena na statistice. Pro nový vzorek dat určuje pravděpodobnost, do jaké třídy bude vzorek patřit. Zařazení do této třídy

bude probíhat tak, že se jednotlivé pravděpodobnosti seřadí podle velikosti a ta s největší pravděpodobností se prohlásí za správnou.

Bayesův vzorec využívá podmíněnou pravděpodobnost. Tu můžeme definovat následovně:

$$P(X|Y) = \frac{P(X \cap Y)}{P(Y)} \tag{2.7}$$

V rovnici 2.7 jsou dva obecně různé jevy X a Y. $P(X|Y)$ značí podmíněnou pravděpodobnost jevu X za podmínky výskytu jevu Y. Tato hodnota tedy udává, jaká je pravděpodobnost, že nastane jev X, pokud víme, že nastal jev Y. $P(X \cap Y)$ je pravděpodobnost, že nastanou jevy X a Y současně. $P(Y)$ je pravděpodobnost jevu Y.

Bayesův vzorec lze snadno odvodit právě od podmíněné pravděpodobnosti a vypadá následovně:

$$P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)} \tag{2.8}$$

Pokud budeme provádět klasifikaci dat na základě jednoduché bayesovské klasifikace, můžeme zavést následující označení:

S – Značí množinu všech vzorků, tedy trénovacích dat.

$C_1, C_2, \ldots, C_m$ – Značí jednotlivé třídy, do kterých jsou vzory z množiny S klasifikovány. Symbol m udává celkový počet těchto tříd.

s_i – Značí počet prvků množiny S, který je klasifikovaný do třídy C_i, kde $i = 1, 2, \ldots, m$.

$A_1, A_2, \ldots, A_n$ – Značí jednotlivé atributy.

$X = (x_1, x_2, \ldots, x_n)$ – Značí testovací vzorek, který má být klasifikován do nějaké třídy, kde x_i je hodnota atributu A_i pro všechna $i = 1, 2, \ldots, n$

Můžeme podle zavedení výše zmíněných konvencí vzorec upravit do následujícího tvaru:

$$P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)} \tag{2.9}$$

$P(C_i|X)$ udává podmíněnou pravděpodobnost toho, že prvek X patří do třídy C_i a najít takové i, pro které je $P(C_i|X)$ největší. To bude tehdy, pokud bude maximální výraz $P(X|C_i)P(C_i)$. $P(C_i)$ je pravděpodobnost, že libovolně zvolený prvek patří do třídy C_i. Tu lze jednoduše určit pomocí vzorce 2.10, $|S|$ značí počet prvků, který tato množina obsahuje.

$$P(C_i) = \frac{s_i}{|S|} \tag{2.10}$$

$P(X|C_i)$ je naopak pravděpodobnost, že libovolný prvek vybraný ze třídy C_i bude mít stejné hodnoty jako prvek X. Tato pravděpodobnost se získá jako součin pravděpodobností $P(x_k|C_i)$ pro $k = 1, .., n$, kde $P(x_k|C_i)$ je pravděpodobnost, že prvek, jehož atribut A_k je roven hodnotě x_k bude zařazen do třídy C_i. Můžeme tuto skutečnost zapsat následovně 2.11:

$$P(X|C_i) = \prod_{k=1}^{n} P(x_k|C_i) \tag{2.11}$$

Hodnoty $P(x_k|C_i)$ se určují podle typu atributu A_k. Jestliže má atribut diskrétní charakter, potom lze napsat 2.12, kde s_{ik} je počet vzorků z množiny S zařazené do třídy C_i, jejichž atribut A_k má hodnotu x_k.

$$P(x_k|C_i) = \frac{s_{ik}}{s_i} \tag{2.12}$$

Pro atribut, který má spojitý charakter se využívá Gaussovo normální rozložení g 2.13 pro atribut A_k s hodnotou x_k. Dále jsou zde parametry průměrné hodnoty μ_{C_i} a směrodatné odchylky σ_{C_i} pro atributy A_k patřící do třídy C_i.

$$P(x_k|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i}) = \frac{1}{\sqrt{2\pi\sigma_{C_i}}} e^{-\frac{(x_k - \mu_{C_i})^2}{2\sigma_{C_i}^2}} \tag{2.13}$$

Další klasifikátory

Zde je uveden stručný popis dalších klasifikátorů, se kterými bylo experimentováno v rámci úlohy č. 2 v kapitole 3.4.

- **Gradient Boosted Tree** – Tato metoda strojového učení využívá kombinace slabých prediktorů (rozhodovacích stromů) omezené velikosti.
- **Deep Learning** – Metoda využívající hlubokou neuronovou síť.

2.3.4 Shluková analýza

Shluková analýza je dalším typem dolovacích úloh. Na rozdíl od klasifikace a predikce shlukování analyzuje datové objekty bez znalosti přiřazení do tříd. Hlavním cílem shlukování je nalézt třídy objektů, které mají co nejvíce společných vlastností a naopak rozlišit ty, které se od nich liší. Objekty se shlukují do tříd na principu maximalizace podobnosti objektů téže třídy a minimalizace podobnosti objektů různých tříd. Třídy takto vytvořené mají podobu shluků.

Shlukovou analýzu lze aplikovat na data, u kterých chceme například identifikovat homogenní skupiny zákazníků. Tato informace se potom dá použít na cílení reklamní kampaně pro skupiny konkrétních zákazníků. Shluková analýza se dá použít i v případech, kdy naopak hledáme data, která se vyčleňují a odlišují od ostatních. Takovým datovým objektům se říká odlehlé objekty a jejich hledání se nazývá analýza/dolování odlehlých objektů.

Hlavním požadavkem, který je kladený na shlukovací metody, je schopnost efektivního zpracování velkých objemů dat. Shlukovací metody by měly splňovat určité vlastnosti. První z nich je škálovatelnost, která klade důraz na zpracování z pohledu velikosti dat a také na to, aby algoritmy dobře pracovaly s různě velkými objemy dat. Mnohé aplikace vyžadují shlukování dat různého typu, proto další vlastností by mělo být pracovat nejen s numerickými daty, ale i s binárními či kategorickými daty. Shluky mohou mít různou hustotu a tvar. Proto je požadována schopnost vytváření shluků libovolného tvaru a umožnit tak lépe reprezentovat dané třídy.

Mnoho shlukovacích metod vyžaduje zadávání vstupních parametrů. Tyto parametry však mají velký vliv na kvalitu nalezených shluků, nehledě na to že je obtížné nalézt vhodné hodnoty vstupních parametrů. Proto by měly být požadavky na znalost problémů při určování parametrů co nejnižší. Vstupní data mohou obsahovat šum, což vede ke zhoršení kvality výsledných shluků. Shlukovací metody by se tedy měly umět vypořádat se šumem. Shlukovací algoritmus by neměl být citlivý na pořadí vstupních záznamů. Vstupní záznamy obvykle obsahují velké množství atributů. Shluková metoda by měla být schopna zpracovávat datové položky s větším počtem atributů. Aplikace běžně vyžadují shlukování na základě různých omezení. Metody by měly najít třídy dat, které splňují požadovaná omezení. Na závěr je nutné výstup shlukovací metody vhodně a srozumitelně reprezentovat uživatelům.

Existují dvě hlavní datové struktury, se kterými se provádí shlukování:

Matice dat

- Matice dat reprezentuje n objektů. Tyto objekty mohou reprezenovat například produkty, věci, zvířata, apod.
- Každý objekt má m vlastností. Pokud budeme uvažovat jako objekt produkt, tak jeho vlastnost může být jméno, cena, typ, popis apod.
- Matice dat 2.14 je tedy sestrojena z n sloupců a m řádků.

$$\begin{bmatrix} x_{11} & \dots & x_{1m} \\ \vdots & \ddots & \vdots \\ x_{n1} & \dots & x_{nm} \end{bmatrix} \tag{2.14}$$

Matice odlišností

- Obsahuje vztahy dvojic, které reprezentují všechny kombinace jednotlivých objektů.
- Velikost matice je n x n.
- V matici platí komutativita.
- Matice odlišností 2.15 má na hlavní diagonále hodnoty 0.

$$\begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ \vdots & \vdots & \vdots & \ddots & \\ d(n,1) & d(n,2) & \dots & \dots & 0 \end{bmatrix} \tag{2.15}$$

Existuje celá řada shlukovacích metod. Výběr metody závisí na typu dat, která chceme analyzovat, a na konkrétním účelu. Metody založené na rozdělování vyžadují určení počtu tříd. Objekty jsou rozděleny do předem určeného počtu shluků. V prvním kroku shlukování se vybere k náhodných objektů, které jsou reprezentanty jednotlivých tříd, a ostatní prvky se na základě podobnosti (vzdálenostní fukce) k těmto třídám přiřazují. Dále se hledají reprezentanti jednotlivých tříd a objekty se přesouvají. Nevýhoda těchto metod je, že je nutné předem určit počet shluků. Dalším problémem je schopnost hledání shluků různých tvarů a citlivost metod na šum. Mezi nejznámější patří metoda založená na centrálním bodu(*k-means*) a metoda založená na reprezentujícím objektu (*k-medoids*).

Hierarchické metody vytvářejí hierarchický rozklad dané množiny objektů (vzniká strom shluků). Podle toho, jak probíhá tento rozklad, rozlišujeme hierarchické metody na shlukující a rozdělující. Nevýhodou těchto metod je, že jakmile některé třídy sloučíme nebo rozdělíme, již není možné tyto třídy znovu sloučit nebo rozdělit. Metody mohou mít lepší výpočetní složitost než rozdělovací metody, avšak nemusejí být tak přesné. Rozdělení také vyžaduje vyhodnocení velkého počtu různých možností. Mezi nejznámější metody patří například metody *BIRCH*, *ROCK* nebo *Chameleon*.

Metody, které jsou schopné dobře se vypořádat s výskytem šumu, jsou metody založené na hustotě. Tyto metody považují za shluky oblasti, které mají velkou hustotu objektů v prostoru dat a které jsou oddělené oblastmi s malou hustotou vyskytujících se objektů. Oblasti, které mají malou hustotu vyskytujících se objektů, se považují za šum. Mezi nejznámější metody založených na hustotě jsou *DBSCAN* a *DENCLUE*.

Metody založené na mřížce využívají víceúrovňovou datovou strukturu. Prostor objektů rozdělují na konečný počet buněk, které tvoří mřížku a všechny operace jsou prováděny právě nad touto mřížkou. Typickým zástupcem těchto metod je metoda *WaveCluster*, která využívá vlnkovou transformaci původního prostoru dat. Zpracování dat je velmi rychlé, nevyžaduje zadání vstupních parametrů a dobře hledá shluky různých tvarů a umí se vypořádat s odlehlými hodnotami.

Optimalizovat shodu mezi datovou množinou a matematickým modelem se snaží metody založené na modelech. Snaží se nalézt shluky, které co nejvíce odpovídají danému modelu. Metody jsou založeny na principu, že data jsou generována na základě nějaké složené pravděpodobnostní distribuční funkce. Zástupcem těchto metod je například metoda *Expectation – Maximization*, která rozšiřuje algoritmus *k-means*.

Většina shlukovacích metod pracuje s malým počtem dimezí (atributů). Jakmile mají pracovat s vysocedimenzionálními daty, nastává problém. Se zvyšujícím se počtem dimenzí je pouze malé množství dimenzí relevantních pro jednotlivé shluky. V ostatních dimenzích se navíc může nacházet spousta datového šumu. Data v různých dimenzích se mohou považovat za stejně vzdálená a nelze tedy využívat vzdálenostní funkce k hledání shluků. Řešením bývá metoda transformace rysů, která transformuje data do prostoru s menším počtem dimenzí a zachová relativní vzdálenosti mezi objekty. Neodstraňuje žádné atributy, využívá sumarizaci dat vytvářením kombinací atributů. Při velkém počtu irelevantních atributů však tyto atributy mohou maskovat skutečné shluky. Další metodou je výběr atributů, kde už se odstraňují irelevantní dimenze. Nacházejí se tedy atributy, které již jsou relevantní pro danou úlohu a provádí se hledání těchto atributů většinou pomocí strojového učení.

Kapitola 3

Popis dolovacích úloh

Tato kapitola se zabývá v úvodu popisem firemních dat, která mi byla poskytnuta. Dále jsou na základě požadavků a struktury dat vytvořeny úlohy, se kterými bylo experimentováno v prostředí nástroje RapidMiner.

V rámci zkvalitňování a poskytování služeb se firma chce pokusit získat podrobnější informace o svých produktech a zákaznících. Každý zákazník má nějaké požadavky v průběhu roku a firma by rovněž chtěla zjistit, zda i v oblasti požadavků lze najít užitečné informace. Zda mají data mezi sebou nějaké vazby ve vztahu zákazník a produkt, případně zákazníci a jejich požadavky. Neexistuje u nich zatím žádný nástroj, který by jim pomohl v rozhodování, jaký nový produkt by si stávající zákazník mohl zakoupit, případně jaká je pravděpodobnost, že si zákazník, který má určité parametry, koupí daný produkt. Jako parametry zákazníka jsou zde myšleny například počty osobních čísel, počty zakoupených produktů, případně důležitost zákazníka.

V návaznosti na požadavky firmy bych se chtěl pokusit řešit následující úlohy:

1. Nabídka souvisejících služeb.
2. Predikce, zda si zákazník s určitými parametry zakoupí konkrétní produkt.
3. Shluková analýza z dat požadavků od klientů.

3.1 Popis firemních dat

Experimenty budu provádět na reálných datech firmy. Jedná se o data, která evidují veškeré informace o svých zákaznících, jejich zakoupených licencí na konkrétní produkty. Dále data obsahují požadavky od zákazníků na konkrétní produkt (oblast) v systému. Z požadavků následně vzniká úkol, který se ve firmě řeší. Protože se ve firemním informačním systému nachází v každé tabulce velké množství atributů (důležitých pro chod systému), byly vybrány jen relevantní atributy, které jsou považovány za zajímavé a budou sloužit pro dolovací úlohy.

Z důvodu zachování anonymity a ochrany osobních údajů, nejsou u zákazníků uvedeny jejich jména. Zákazník tak vystupuje pouze pod svým číslem. O kterého zákazníka se jedná, lze dohledat pouze interně v informačním systému firmy.

Následuje popis těch atributů, které jsou z tabulek vybrány a jsou považovány za důležité:

Požadavky

Tabulka eviduje všechny požadavky, které přišly od zákazníků.

- **ID požadavku** – je jednoznačný identifikátor každého požadavku.
- **firma** – je jednoznačný číselný identifikátor zákazníka, který požadavek vytvořil.
- **oblast** – jedná se o oblast příslušící danému produktu, do kterého spadá požadavek.
- **evidováno** – je datum, kdy byl požadavek vytvořen.
- **stav** – určuje, v jakém stavu se požadavek nachází a může nabývat následujících hodnot:
  - *0* – nezapsáno
  - *1* – nepřidělelno
  - *2* – přiděleno, nezahájeno
  - *3* – dosud nekontaktován zadavatel
  - *4* – řešeno
  - *5* – pozastaveno
  - *6* – ukončeno bez generování odpovědi
  - *7* – ukončeno s generováním odpovědi
- Data byla vyexportována do souboru `pozadavky.csv`

Produkty

Tabulka eviduje všechny produkty, které firma nabízí. Jsou zde i interní produkty.

- **modul** – je jednoznačný identifikátor každého produktu.
- **oblast** – určuje oblast, do které daný produkt spadá.
- **název** – celý název produktu.
- Data byla vyexportována do souboru `produkty.csv`

Zákazníci

Tabulka obsahuje informace o všech zákaznících (současných i minulých), které firma oslovila nebo kteří jsou jejími aktuálními zákazníky.

- **firma** – je jednoznačný identifikátor každého zákazníka.
- **stav zákazníka** – číselná hodnota, která udává stav zákazníka, může nabývat následujících hodnot:
  - *1* – současný
  - *2* – bývalý
  - *3* – potenciální
  - *4* – minulý zájemce

- **počet pracovníků** – číselná hodnota, která udává, kolik osobních čísel (počet pracovníků) má zákazník zakoupených.
- **důležitost** – číselná hodnota, která udává významnost zákazníka, může nabývat následujících hodnot:
  - *1* – VIP
  - *2* – významný partner
  - *3-7* – skupina 3–7
  - *8* – pracovníci firmy
  - *9* – běžný zákazník
  - *11* – zákazník s omezením
  - *18* – externí zpracovatel
- **aplikace** – udává seznam zkratek produktů, které má zákazník zakoupené.
- Data byla vyexportována do souboru `zakaznici.csv`

Firemní data byla ještě před samotnými experimenty v rámci předzpracování upravena a transformována do požadované podoby. Jakým způsobem byla data upravena je uvedeno u konkrétních úloh.

3.2 RapidMiner

RapidMiner je softwarová platforma, která poskytuje integrované interaktivní prostředí pro zpracování dat, strojové a hloubkové učení, dolování z textu a prediktivní analýzu. Používá se jak pro obchodní, tak i pro komerční účely či výzkum. Tento nástroj umožňuje přípravu dat, jednotlivé kroky procesů, validaci a optimalizaci dat a nakonec i vizualizaci výsledků [14]. RapidMiner je napsán v jazyce Java. Jednotlivé postupy a kroky se zde nazývají procesy a skládají se z jednotlivých operátorů [12]. Každý tento operátor provádí svou konkrétní činnost. Tato činnost může být například výběr podmnožiny dat, transformace hodnot nebo operátor pro vytvoření modelu. Každý operátor má svůj vstup a výstup a výsledek se propojuje s jednotlivými porty.

Základní licence, která je 30 dní zdarma, je limitovaná počtem zpracovávaných řádků a logickým procesorem. Pro účely této práce mi byla poskytnuta akademická licence na 1 rok, která umožňuje mít již neomezený počet zpracovávaných řádků, dále automatický učící model, který urychluje výpočet učení a modul pro lepší předzpracování a přípravu dat.

RapidMiner byl použit na experimentování s firemními daty a byl použit ve všech třech úlohách, které zjišťují úspěšnost jednotlivých dolovacích metod.

3.3 Úloha 1: Nabídka souvisejících služeb

První úloha, která bude pro firmu velkým přínosem, je zjištění souvislosti mezi zakoupenými produkty. Vychází z analýzy nákupního košíku a dolování asociačních pravidel.

V rámci předzpracování dat byla data z tabulek produktů a zákazníků zpracována v jazyce Python do výsledné podoby následovně:

Tabulka produktů

- Z tabulky produktů (soubor `produkty.csv`) byly zjištěny všechny zkratky produktů. Tento seznam zkratek potom slouží jako klíč bitového vektoru.

Tabulka zákazníků

- Z tabulky zákazníků (soubor `zakaznici.csv`) byl při procházení kontrolován sloupec *licence*, který obsahuje seznam zkratek produktů oddělených čárkou.
- Na základě tohoto seznamu byl naplněn vektor produktů, který byl získán z tabulky produktů.
- Bitový vektor byl naplněn následujícím způsobem:

  1 – reprezentuje, že daný produkt má zákazník zakoupen

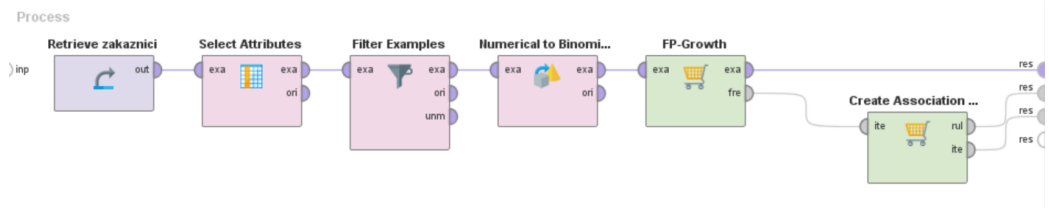
  0 – v ostatních případech

- Dále byly ponechány parametry zákazníka, jako počet pracovníků, stav zákazníka a důležitost.
- Byly vypočteny nové atributy:

  Počet zakoupených produktů – reprezentuje celkový počet produktů, které má zákazník zakoupeny

  Počet portálových aplikací – portálové aplikace začínají ve zkratce produktu malým písmenem *e* a většinou se dodávají zároveň s hlavními produkty

Výsledný soubor zákazníků obsahuje tedy parametry zákazníka a navíc sloupce zkratek všech produktů. Dolování asociačních pravidel díky takto předpřipraveným datům už je pak jednoduché.



Obrázek 3.1: Vytvoření asociačních pravidel

Na obrázku 3.1 je zobrazeno, které operátory byly využity v nástroji RapidMiner. Z načtených dat mohou být vybrány jen konkrétní množiny produků, se kterými se chce provádět experiment. K výběru atributů slouží operátor `Select Attributes`. V rámci této úlohy jsem vybral pouze atributy `zakaznik` (udávající číslo zákazníka) a potom všechny sloupce s produkty. Parametry zákazníka zde nebyly uvažovány a byly odstraněny.

Typ atributů u všech produktů je přetypován na hodnoty `True` a `False`. Přetypování se provádí dle nul a jedniček pomocí operátoru `Numerical to Binominal`. Přetypování potom slouží jako vstup pro algoritmus `FP-Growth`, který vypočítá nejvíce frekventované položky z dané množiny dat.

Minimální podpora byla nastavena na 0.4. Tedy ve 40% všech analyzovaných dat by se měly vyskytovat produkty společně. Hodnota spolehlivosti byla nastavena na 0.8.

Výsledkem je vytvoření asociačních pravidel pomocí operátoru `Create Association rules`, ze kterých je možné získat znalost, jaký vztah mají mezi sebou jednotlivé produkty. Rovněž se ve výsledku dolování nacházejí i frekventované množiny.

V příloze v tabulkách A.1 a A.2 je možné vidět výčet některých frekventovaných množin. Vždy se jedná o množiny dané velikosti, u kterých je výsledná hodnota podpory. V tabulce A.1 je možné ze začátku vidět například jednoprvkové množiny. Čtyři produkty se vyskytují v mnohonásobně vyšší míře než ostatní. Jejich hodnota podpory je u prvních tří produktů více než 86% a u čtvrtého produktu 65%. Jedná se o produkty *PAM* – Mzdy, *ELD* – Elektronické podání, *RNP* – Registrace nemocenského pojištění a *HB** – Převodní příkazy bank. Další produkty jsou oproti těmto čtyřem zastoupeny v nižší míře. Jsou to produkty *KZP* – Komunikace se zdravotní pojišťovnou, *PER* – personalistika, *UCT* – účetnictví, *FAK* – fakturace a další. Například šetiprvková množina v tabulce A.1 má pak podporu 13% a obsahuje také nejčetnější produkty.

V rámci experimentu bylo nastaveno, aby frekventované množiny obsahovaly produkt *VZD* – vzdělávání. Je to z toho důvodu, že tento produkt není zastoupen tak v hojné míře. Frekventované množiny, které právě obsahují tento produkt, je možné vidět v tabulce A.2. Je vidět, že podpora tohoto produktu ze všech analyzovaných dat, je vůči ostatním produktům velice nízká. Produkt *VZD* sám o sobě má podporu 3.1%. V kombinaci s nejčetnějšími produkty je hodnota podpory obdobná, avšak s některými produkty je podpora kolem 1%. Produkt *VZD* bude dále zahrnut v klasifikačních metodách v kapitole 3.4 a budou na něm testovány jednotlivé klasifikační metody.

Předpoklad	Závěr	Podpora	Spolehlivost
ELD, RNP	PAM	0.868	1
RNP, PER	PAM	0.248	1
RNP, HB*	ELD	0.615	1
ELD, RNP, HB*, KZP, PER	PAM	0.134	1
PER	PAM, EKD,RNP, HB*	0.215	0.855
PAM, UCT	FAK	0.113	0.799
UCT	PAM, FAK	0.113	0.746
⋮	⋮	⋮	⋮
VZD	HB*, KZP	0.024	0.793
PAM, VZD	PAM, HB*, KZP	0.024	0.793
RNP, KZP, PER, VZD	PAM	0.025	1

Tabulka 3.1: Asociační pravidla nejčetnějších produktů

Předpoklad	Závěr	Podpora	Spolehlivost
HB*, VZD	HN0018	0.012	1
VZD, ePAM	STM, ePER	0.009	0.850
HF0009, HN0018, VZD	STM	0.008	0.750
HF0009, VZD	HN0018	0.011	0.741

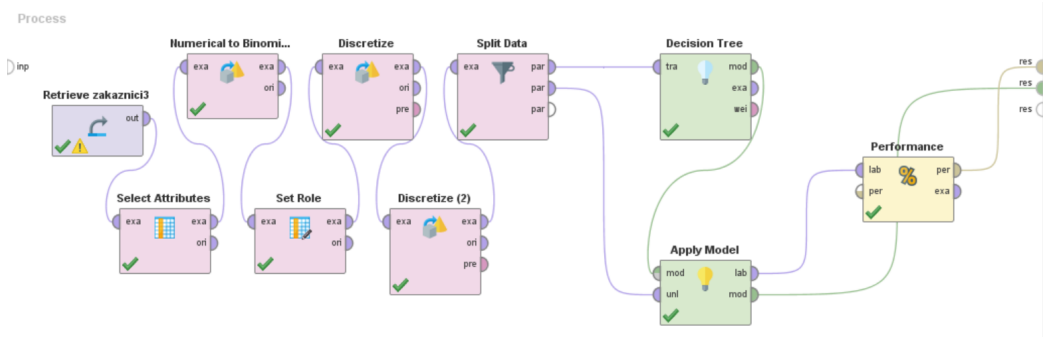
Tabulka 3.2: Asociační pravidla bez nejčetnějších produktů

Nalezená asociační pravidla většinou obsahovala tyto čtyři nejčetnější produkty v různých kombinacích. To ukazuje tabulka 3.1, která obsahuje jen výčet některých získaných asociačních pravidel. Hodnoty podpory byly u produktů, které se vyskytují méně často, nižší (kolem $11-20\%$), avšak spolehlivost byla klidně $80-100\%$. Není žádným překvapením, že se produkt *VZD* vyskytuje s těmito nejčetnějšími produkty společně. Jelikož nemá produkt *VZD* takové zastoupení, je i jeho podpora nízká 2.4%. Nicméně pokud zákazník vlastní produkt *VZD*, je vysoká pravděpodobnost, že by si koupil i některý z nejčetnějších produktů.

Jak již bylo zmíněno, zaměřil jsem se na produkt *VZD*. Odstranil jsem z dat nejfrekventovanější produkty a změnil hodnoty podpory a spolehlivosti. Pak je možné získat další asociace tohoto produktu s ostatními produkty. Výčet některých frekventovaných množin je možné vidět v tabulce A.2. Produkt *VZD* je dále ve spojitosti s produkty *STM* – Systemizace a *HB** – Převodní příkazy bank. Dále s moduly *HN0018* – Konfigurace personálních událostí, *HX0048* – Prověření zaměstnanců v insolvenčním rejstříku a *HF0009* – Penzijní fondy. Hodnota podpory je v průměru kolem 1.5%. Získaná asociační pravidla bez nejčetnějších produktů je možné vidět v tabulce 3.2. Zde se hodnoty podpory přibližují k 1% a hodnoty spolehlivosti jsou nad 74%. Množiny pravidel (tedy seznamy souvisejících produktů), pak posloužily pro druhou dolovací úlohu.

3.4 Úloha 2: Predikce zákazníka

V této úloze, jak již bylo zmíněno výše, byl vytipován konkrétní produkt – produkt *VZD*. Kromě tohoto produktu byla dále vybrána podmnožina produktů (kolem 10ti produktů). V rámci experimentu jsem zkoumal, jestli si zákazník daný produkt koupí nebo ne. Navíc byly brány v úvahu ještě parametry zákazníků – tedy celkový počet zakoupených produktů, stav zákazníka, počet pracovníků, důležitost zákazníka a počet portálových aplikací. Na základě toho jsem vytvořil model a poté experimentoval s různými klasifikačními metodami. Zjišťoval jsem, jaké dávají výsledky pro tento méně frekventovaný produkt. Výsledkem úlohy je informace, zda si zákazník s těmito konkrétními parametry koupí daný produkt.



Obrázek 3.2: Predikce pro konkrétní produkt s využitím operátorů

Z načtených dat zákazníků jsem nejprve zkoušel vybrat jen některé podmnožiny produktů na základě získaných asociačních pravidel. Potom jsem ponechal všechny produkty a zkoumal daný produkt vůči nim. Které operátory byly použity v nástroji RapidMiner, je možné vidět na obrázku 3.2. Jelikož jsem použil stejnou předzpracovanou tabulku zákazníků (předzpracování je popsáno v úloze č.1 3.3), tak obdobným způsobem, jako u aso-

ciačních pravidel, se hodnoty 0 a 1 převedly pomocí operátoru `Numerical to Binomical` na hodnoty `True` a `False`. Operátorem `Set Role` byl nastaven zkoumaný produkt na roli `label` a číslo zákazníka na roli `id`.

V dalších dvou krocích byly parametry zákazníků diskretizovány do patřičných intervalů. Jedná se o atribut *počet pracovníků*, který byl rozdělen na 12 intervalů. Obdobně atribut *počet zakoupených produktů* byl rozdělen na 5 intervalů. Následně byla data rozdělena pomocí operátoru *Split Data*, kde 70% dat sloužilo pro trénování modelu a zbylých 30% na testování modelu.

Byl zvolen zamíchaný vzorek dat. Následně byly zkoumány jednotlivé klasifikační metody. Pro validaci dat byl model aplikovaný pomocí operátoru `Apply Model` a přesnost byla měřena pomocí `Performace` operátoru. Matice záměn nám ve výsledku udává, kdy se klasifikátor shoduje s učitelem a v kolika případech se dopustí chyby. Zaměřoval jsem se tedy na ty metody, které měly nejlepší hodnoty v matici v kategorii `True True`.

3.4.1 Decision Tree

Jako první byla vyzkoušena technika rozhodovacích stromů, o které jsem se zmínil v kapitole 2.3.3. Parametry rozhodovacího stromu jsem nechal nastaveny ve výchozím stavu, jak byly přednastaveny v RapidMineru. Tedy hloubka 10, spolehlivost 0.1 a minimal gain na 0.01. Experimenty s jinými hodnotami nevedly k lepším výsledkům. Pokud jsem vzal v úvahu všechny produkty 3.4, průměry tříd vedly k lepším výsledkům než u vybrané podmnožiny zkoumaných produktů 3.3. Je to z toho důvodu, že menší podmnožina produktů je mnohem striktnější a produkt *VZD* se v datech nevyskytoval tak často. Proto jej metoda zařadila do kategorie `True Ture` pouze v 5 případech.

accuracy: 97.18%

	true false	true true	class precision
pred. false	546	16	97.15%
pred. true	0	5	100.00%
class recall	100.00%	23.81%	

Obrázek 3.3: Přesnost metody Decision tree vůči vybrané podmnožině produktů

accuracy: 98.06%

	true false	true true	class precision
pred. false	545	10	98.20%
pred. true	1	11	91.67%
class recall	99.82%	52.38%	

Obrázek 3.4: Přesnost metody Decision tree vůči všem produktům

Pro srovnání jsem ještě vyzkoušel nejčastnější produkt – produkt *PAM*. Výsledky dopadly tak, jak bychom očekávali (obrázek 3.5). Klasifikátor zařadil z trénovacích dat 549 hodnot do kategorie `True True`.

accuracy: 97.71%

	true false	true true	class precision
pred. false	5	6	45.45%
pred. true	7	549	98.74%
class recall	41.67%	98.92%	

Obrázek 3.5: Přesnost metody Decision tree nejčetnějšího produktu PAM

3.4.2 Naive Bayes

Jako další metoda pro experimentování byla zvolena metoda `Naive Bayes`. Tato metoda využívá Bayesovskou klasifikaci popsanou v kapitole 2.3.3. Je to tedy statistická metoda, která vyjadřuje jistotu, s jakou byla data správně klasifikována. Pro rozdělení vstupních dat bylo opět použito zamíchání. Tato metoda dává pro vybranou množinu produktů 3.6 lepší výsledky než metoda `Decision Tree`.

accuracy: 96.65%

	true false	true true	class precision
pred. false	534	7	98.71%
pred. true	12	14	53.85%
class recall	97.80%	66.67%	

Obrázek 3.6: Přesnost metody Naive Bayes vůči vybrané podmnožině produktů

accuracy: 96.47%

	true 0	true 1	class precision
pred. 0	534	8	98.52%
pred. 1	12	13	52.00%
class recall	97.80%	61.90%	

Obrázek 3.7: Přesnost metody Naive Bayes vůči všem produktům

3.4.3 Gradient Boosted Tree

Tato metoda strojového učení využívá kombinace slabých prediktorů (rozhodovacích stromů) omezené velikosti. Parametry pro metodu `Gradient Boosted Tree` byly nastaveny následovně: počet stromů 20, hloubka 20, min. řádků 10, počet košů 20, learning rate na nejnižší hodnotu 0.1, sample rate 1.0 a rozložení `bernoulli`, které je vhodné právě pro binominální hodnoty.

accuracy: 97.00%

	true false	true true	class precision
pred. false	535	6	98.89%
pred. true	11	15	57.69%
class recall	97.99%	71.43%	

Obrázek 3.8: Přesnost metody Gradient Boosted Tree vůči vybrané podmnožině produktů

accuracy: 96.12%

	true false	true true	class precision
pred. false	532	8	98.52%
pred. true	14	13	48.15%
class recall	97.44%	61.90%	

Obrázek 3.9: Přesnost metody Gradient Boosted Tree vůči všem produktům

3.4.4 ID3

ID3 je algoritmus pro konstrukci rozhodovacího stromu, který využívá kritéria pro výběr atributů do uzlů stromu, které odpovídají i kritériím pro výběr atributů. Kritérium bylo nastaveno `gain ratio`, minimální velikost rozdělení 2, minimální velikost listu 1 a minimální gain 0.1.

accuracy: 97.00%

	true 0	true 1	class precision
pred. 0	541	12	97.83%
pred. 1	5	9	64.29%
class recall	99.08%	42.86%	

Obrázek 3.10: Přesnost metody ID3 vůči vybrané podmnožině produktů

accuracy: 97.53%

	true 0	true 1	class precision
pred. 0	539	7	98.72%
pred. 1	7	14	66.67%
class recall	98.72%	66.67%	

Obrázek 3.11: Přesnost metody ID3 vůči všem produktům

3.4.5 Deep Learning

Pro hlubokou neuronovou síť jsem data zkoušel rozdělit lineárně, zamíchaně i automaticky. Výsledek se ale vždy lišil. Když jsem pro konkrétní rozdělení model spouštěl vícekrát, pokaždé dával jiné hodnoty v kategorii `True True`. Jednou to bylo kolem 80% jindy zase kolem 30%. Je tedy důležité, na kterých konkrétních datech se síť učí. Co se týká parametrů, tak pro aktivační funkci dávaly nejlepší výsledky funkce `Tanh` a `Rectifier`. Pro všechny produkty bylo vhodnější použít `Tanh`. Počet iterací dat byl nastaven na hodnotu 10.0. Skryté vrstvy byly nastaveny na 50 na 50.

accuracy: 94.89%

	true 0	true 1	class precision
pred. 0	530	3	99.44%
pred. 1	26	8	23.53%
class recall	95.32%	72.73%	

Obrázek 3.12: Přesnost metody Deep Learning vůči vybrané podmnožině produktů

accuracy: 96.12%

	true false	true true	class precision
pred. false	527	3	99.43%
pred. true	19	18	48.65%
class recall	96.52%	85.71%	

Obrázek 3.13: Přesnost metody Deep Learning vůči všem produktům

V této úloze byl zvolen produkt *VZD*, který není tak četný. Byly na něm vyzkoušeny různé klasifikační metody. Produkt byl zkoumán vůči vybrané podmnožině produktů, ale i vůči všem ostatním produktům. Zkoumal jsem, jak se dané metody chovají a jaké dávají výsledky. Vybraná podmožina obsahovala všechny čtyři nejčastěji se vyskytující produkty. Dále množina obsahovala i produkty, které byly zjištěny z frekventovaných množin z úlohy 3.3. Je logické, že by ve vybrané množině produktů měly být zahrnuty i nejčetnější produkty. Firma z nich má totiž největší zisk. Tím, že data ze souboru obsahují jen hodnoty reprezentující koupeno/nekoupeno, případně byly ještě diskretizovány atributy do společných intervalů, dávaly skoro všechny klasifikační metody dobrou přesnost, v průměru kolem 96%. Proto jsem se zaměřil na hodnoty `True True` (případně 1 1) a na ty metody, které dávaly nejlepší zařazení pro tuto kombinaci a měly co nejnižší počet špatného zařazení do dané třídy. Nejlepší výsledky dávaly metody Gradient Boosted tree, Naive Bayes a Deep learning.

3.5 Úloha 3: Shluková analýza z dat požadavků od klientů.

Poslední úloha spadá do oblasti shlukování. Firma eviduje od svých zákazníků požadavky, ze kterých potom vzniká úkol, který se řeší. Metoda shlukování, jak je uvedeno v kapitole 2.3.4, slouží k přiřazení datových objektů do skupin. Objekty se snaží zařadit podle společných vlastností a naopak rozlišit ty, které se od ostatních co nejvíce liší. V této úloze právě využiji analýzu odlehlých hodnot – tedy budu hledat ty zákazníky, kteří se nejvíce odlišují od ostatních. Nalezení takto odlehlých hodnot (zákazníků) je pro firmu cenná informace, kterou může využít při rozhodování, kterým zákazníkům by se měla více věnovat na základě jejich požadavků.

Tento experiment pracuje s tabulkou požadavků získanou z informačního systému firmy. Data jsou popsána v kapitole 3.1. Jelikož pro tuto úlohu nejsou relevantní atributy `ID požadavku` a `stav` (všechny požadavky od roku 2013 již byly vyřešeny), byly z výsledného souboru odstraněny.

Data byla rovněž v rámci předzpracování upravena v jazyce Python do následující podoby:

Tabulka požadavků

- Tabulka obsahuje zánamy požadavků od roku 2013 a časový údaj, od kterého zákazníka byl požadavek přijat a z které oblasti.
- Časové období bylo rozděleno na kvartály jednotlivého roku.
- Pro každého zákazníka byl vypočítán počet požadavků v daném kvartálu v dané oblasti a nový atribut udávající celkový počet požadavků.
- Výsledná tabulka tedy obsahuje sloupce:

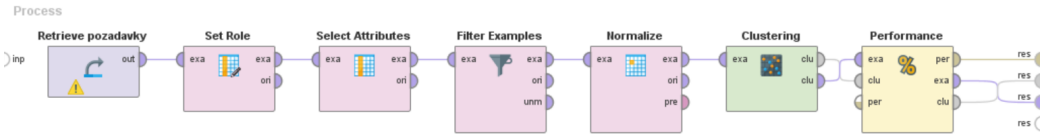
  zakazník – identifikátor zákazníka.

  oblast – oblast, ve které měl zákazník někdy nějaký požadavek.

  celkový počet požadavků – celkový počet požadavků od roku 2013 z dané oblasti.

  kvartály – jednotlivé kvartály každého roku.

V nástroji RapidMiner 3.14 byly vybrány sloupce za poslední rok operátorem `Select Attributes`. V dalším kroku byly atributy normalizovány pro shlukovací metodu `K-means`.



Obrázek 3.14: Přesnost metody Deep Learning vůči všem produktům

V rámci experimentu jsem nejprve vyzkoušel nechat všechny řádky. V datech se tedy objevovali i ti zákazníci, kteří nemuseli mít žádný požadavek v dané oblasti za celý rok. Následně jsem nastavil filtr tak, aby obsahoval pouze ty zákazníky, kteří měli v každém kvartálu alespoň jeden požadavek.

U metody `K-means` je nutné nastavit počet shluků. Hodnota počtu shluků se velice špatně odhaduje. Proto po několika experimentech vyšlo nejlépe pro všechny kvartály zvolit počet shluků na hodnotu 15. Dále je vhodné použití filtru dat, který vybere pouze nenulové požadavky a kvartály jen za poslední rok. Počet výsledných shluků je pak vhodné nastavit na hodnotu 8.

Shluk	Počet	Shluk	Počet
0	1228	0	43
1	1	1	2
2	32	2	240
3	9	3	1
4	276	4	8
5	11617	5	112
6	531	6	1
7	710	7	84
8	151		
9	1204		
10	35		
11	1		
12	230		
13	135		
14	2		

Tabulka 3.3: Nalezené shluky a počet zákazníků v každém shluku pomocí metody K-means

Výsledné shluky je možné vidět v tabulce 3.3. V levé části tabulky se nachází 15 výsledných shluků. Výsledek je pro první možnost, kdy nebyl použit žádný filtr dat. Z nalezených shluků jsou zajímavé ty, které mají číslo 1, 3, 11 a 14. Celkový počet řádků, se kterými metoda pracovala, byl 16 162. Následující popis uvádí shluky, které obsahují odlehlé hodnoty. V každém shluku je pak popsáno, co obsahuje za hodnoty a co můžeme o zákaznících prohlásit.

Shluk 1

- O tomto zákazníkovi můžeme prohlásit, že má nejvyšší celkový počet požadavků ze všech (tato hodnota je brána od roku 2013) a zároveň že měl v každém kvartále často požadavky v oblasti *PAM* (v průměru kolem 30 požadavků).
- Můžeme o něm tedy prohlásit, že v této oblasti by se mu měla firma více věnovat, případně se podívat na typy jeho požadavků.

Shluk 3

- V tomto shluku se nacházejí zákazníci, kteří mají požadavky z různých oblastí *PAM*, *PER*, *DCH* a *POK*. Někteří zákazníci neměli vůbec žádné požadavky ve všech kvartálech, případně do 5 požadavků za jeden konkrétní kvartál.
- Hodnota atributu celkového počtu požadavků byla rovněž různá. U těch zákazníků, kteří měli nějaké požadavky, byla vyšší než 100. U těch, kteří neměli žádné nebo jen málo, byla tato hodnota mezi 10-35.
- Tento shluk tedy moc nepomohl v rozhodování, jaké jsou společné znaky těchto zařazených zákazníků.

Shluk 11

- Tento shluk obsahuje opět jen jednoho zákazníka. Celkový počet požadavků 210 měl v oblasti *PAM*. V prvních dvou kvartálech měl 63 požadavků v této oblasti a ve zbylých dvou kvartálech přes 30 požadavků.

– Můžeme o něm také prohlásit, že v této oblasti by se mu měla firma více věnovat, případně se podívat na typy jeho požadavků.

Shluk 14

– Tento shluk obsahuje dva zákazníky. První zákazník je irelevantní, protože neměl žádný požadavek v celém roce.

– Druhý zákazník měl požadavky z oblasti *PAM*, nicméně nebyly vůbec četné – max. 4 požadavky. Proč se ale nachází v tomto shluku, je z toho důvodu, že jeho atribut celkového počtu požadavků v této oblasti je vysoký – 769.

– Můžeme tedy usoudit, že nyní zákazník nemá tolik požadavků, ale je potenciálním adeptem, že by v budoucnu další požadavky mohl mít.

V pravé části tabulky 3.3 je možné vidět výsledné shluky, u kterých byl použit filtr na nenulový počet požadavků v každém kvartále roku. Jak již bylo zmíněno výše, bylo zvoleno 8 výsledných shluků. Potenciálně zajímavé jsou shluky číslo 1, 3, 4 a 6, protože také obsahují odlehlé hodnoty. Díky použití striktního filtru tak, aby obsahoval nenulový počet požadavků ve všech kvartálech, se však celkový počet řádků tabulky snížil na pouhých 491 záznamů.

Shluk 1

– První shluk obsahuje 2 zákazníky, z nichž jeden je stejný jako ve shluku 1 předešlého příkladu. Oba tito zákazníci mají nejvyšší počet všech požadavků ve své oblasti.

– Rovněž i jejich požadavky v jednotlivých kvartálech roku jsou vyšší než obvykle. Můžeme tedy prohlásit tyto zákazníky na adepty, kterým by se měla firma pravděpodobně více věnovat.

Shluk 2

– Shluk 2 obsahuje stejného zákazníka jako shluk 11 v předchozím příkladu.

Shluk 4

– Tento shluk obsahuje 8 zákazníků. Zákazníci se vyznačují tím, že mají buď stejnou oblast a přibližně i stejný počet požadavků, nebo stejný počet požadavků ve vícero kvartálech roku.

– Dále tito zákaníci mají celkový počet požadavků vyšší než 188. Polovina těchto zákazníků spadá opět do oblasti *PAM*. Tři zákazníci se zároveň nacházeli ve shluku číslo 3 z předchozího příkladu.

Shluk 6

– Shluk obsahuje pouze jednoho zákazníka. Jedná se o stejného zákazníka, který se nachází ve shluku č. 14 z předešlého příkladu.

Metodu shlukování je možné shrnout následovně. Pro lepší výsledky je vhodnější použít filtr na data pro konkrétní období na nenulové hodnoty v jednotlivých kvartálech, případně

ještě více zúžit interval roku. Pokud tak neučiníme, metoda sice dobře odhalí zákazníky, kteří se odlišují tím, že nemají žádné požadavky, ale tato informace je pro firmu zcela irelevantní. Zákazník, který nemá požadavky, je spokojený zákazník. Po použití filtru metoda nalezla většinou zákazníky, kteří mají hodně požadavků nebo požadavky závisely na atributu celkového počtu požadavků. Pravděpodobně by se dostalo jiných výsledků, kdyby se atribut celkového počtu požadavků počítal dynamicky dle vybraných sloupců jednotlivých kvartálů. Výsledkem této úlohy je několik potenciálních klientů, na které by se měla firma zaměřit a případně se jim měla více věnovat.

Kapitola 4

Použité technologie

Na začátku tato kapitola popisuje technologie pro vývoj a dále zvolený nástroj Weka. Tento nástroj byl zvolen z toho důvodu, že jeho API[1] je napsáno v jazyce Java. Dalším důvodem bylo to, že má velice dobrou dokumentaci a existuje k němu hodně tutoriálů, ve kterých je ukázáno, jak je možné nástroj používat pro vlastní aplikace. Navíc je to velice dobrý nástroj, který obsahuje knihovny nejen pro předzpracování dat, ale i spoustu algoritmů a nástrojů pro samotné strojové účení a predikce. Proto byl nástroj Weka použit pro implementaci výsledné aplikace.

Dále se tato kapitola zabývá popisem vstupních souborů ve formátu ARFF a popisem významných tříd a balíků Weky, které byly použity v rámci řešení.

4.1 Technologie pro vývoj

V této podkapitole se jen stručně zmíním o použitém programovacím jazyce Java a zvoleném vývojovém prostředí, které bylo použito pro samotnou implementaci a testování výsledné aplikace. Popis Javy, její specifikace a odlišnosti od jiných programovacích jazyků vycházejí z [9]. Informace o vývojovém prostředí NetBeans vycházejí z [11].

4.1.1 Jazyk Java

Java je objektově orientovaný programovací jazyk, který vyvinula firma Sun Microsystems v roce 1995. Jde o jeden z nejpoužívanějších programovacích jazyků na světě. Díky své přenositelnosti je používán pro programy, které mají pracovat na různých systémech. Může jít o čipové karty (platforma JavaCard), aplikace pro mobilní telefony a různá zabudovaná zařízení (platforma Java ME), aplikace pro desktopové počítače (platforma Java SE) či rozsáhlé distribuované systémy (platforma Java EE). Tyto technologie se jako celek nazývají platforma Java. Syntaxe Javy je zjednodušenou verzí syntaxe jazyka C a C++. Java nepodporuje přetěžování operátorů nebo vícenásobnou dědičnost pro třídy. To zjednodušuje jazyk a pomáhá při prevenci potenciálních chyb. Nejsou zde ukazatelé, bezznaménkové číselné datové typy, příkaz goto nebo preprocesor. S výjimkou osmi primitivních datových typů (tj. celá čísla, desetinná čísla, logické hodnoty a znaky) jsou všechny ostatní datové typy objektové.

Java je navržena pro vytváření aplikací v síti, distribuovaných klientských aplikací a serverů. Program může pracovat na libovolném počítači, který má k dispozici interpret Javy –

[1] API - Application Programming Interface.

tzv. virtuální stroj Javy (JVM[2]). Java je určena pro psaní vysoce spolehlivého softwaru. Z tohoto důvodu Java neumožňuje některé programátorské konstrukce, které bývají častou příčinou chyb. Jedná se například o správu paměti, příkaz goto, ukazatele. Java používá silnou typovou kontrolu, tedy všechny defiované proměnné musejí mít svůj datový typ.

Správa paměti je realizována pomocí automatického garbage collectoru, který automaticky vyhledává již nepoužité části paměti a uvolňuje je pro další použití. Programátor určuje, kdy bude objekt vytvořen a Java zodpovídá za obnovu paměti jakmile se objekty přestanou používat. Když nezůstanou žádné odkazy na objekt, nepřístupná paměť se stává přístupnou pro garbage collector, který ji automaticky uvolní. Zpočátku použití garbage collectoru vedlo k pomalejšímu běhu programů, avšak v posledních verzích je díky používání nových algoritmů pro garbage collector a generační správy paměti tento problém ze značné části eliminován. Generační správa paměti znamená, že paměť je rozdělena na více částí a v každé se používá jiný algoritmus pro garbage collector a objekty jsou mezi těmito částmi přesouvány podle délky svého života.

Java má rovněž prvky, které chrání počítač v síťovém prostředí před nebezpečnými operacemi nebo napadení systému cizím kódem. Aplikace vytvořená v Javě běží na libovolném operačním systému či architektuře. Ke spuštění je pouze nutné mít nainstalovaný virtuální stroj. Jazyk je nezávislý i co se týká vlastností základních datových typů. Je explicitně určena vlastnost a velikost každého z primitivních datových typů.

Přestože se jedná o jazyk interpretovaný, není ztráta výkonu významná. Překladače totiž mohou pracovat v režimu "just-in-time"a do strojového kódu se překládá jen ten kód, který je opravdu zapotřebí. Java samozřejmě podporuje používání vícevláknových aplikací. Knihovna může být za běhu dynamicky rozšiřována o nové třídy a funkce a to jak z externích zdrojů, tak i vlastním programem. Cílem Javy je, aby byly všechny její implementace kompatibilní.

4.1.2 NetBeans

NetBeans je integrované vývojové prostředí (IDE), které vlastní firma Oracle. V současné době je NetBeans v režii Apache Software Foundation. Primárně je toto vývojové prostředí určeno pro vývoj aplikací psaných v Javě. Umožňuje však programování i v jiných programovacích jazycích jako je PHP, HTML5/CSS, JavaScript, C/C++, Groovy. Celé prostředí je rovněž naprogramováno v Javě, takže jej lze spustit na různých operečních systémech. NetBeans nabízí širokou škálu nejrůznějších rozšíření, obsahuje grafický debugger, automatizované nástroje jako je Ant, Maven či Gradel, nechybí syntaktická analýza zdrojového kódu a nápověda. Není to tedy jen obyčejný textový edotor. Umožňuje také zobrazení dokumentace JavaDoc a obsahuje i nástroj pro tvorbu grafického rozhraní pro sestavení aplikačních a dialogových oken. Podporuje rovněž komponenty knihoven AWT a Swing. NetBeans je volně dostupný a je šířen zdarma.

4.2 Java nástroje a knihovny pro strojové učení

Tato sekce obsahuje popis nejpopulárnějších nástrojů a knihoven pro strojové učení. Všechny nástroje jsou psány v jazyce Java. [3]

[2] JVM - Java Virtual Machine.

Weka

Weka má kolekci algoritmů strojového učení pro úlohy dolování dat. Více o tomto nástroji bude popsáno dále v kapitole 4.3.

Rapid Miner

Rapid Miner poskytuje GUI a Java API pro vývoj vlastních aplikací. Nástroj byl použit v rámci experimentálních úloh popsaných v 3.2.

ELKI

ELKI nebo-li Environment for Developing KDD-Applications Supported by Index-Structure je volně dostupný software pro dolování dat napsaný v jazyce Java. ELKI se zaměřuje na výzkum v oblasti algoritmů s důrazem na unsupervised metody v oblasti shlukování a detekci odlehlých hodnot.

MOA

MOA nebo-li Massive Online Analysis je rovněž volně dostupný framework pro dolování datových proudů s velmi rostoucí komunitou. Zahrnuje kolekci algoritmů strojového učení (klasifikace, regrese, shlukování aj.). Zahrnuje v sobě rovněž nástroje pro hodnocení. Je napsán v jazyce Java a postupně se rozšiřuje pro řešení náročnějších problémů.

Apache SAMOA

Je framework pro strojové učení, který obsahuje programovací abstrakci pro distribuované streamingové algoritmy a rovněž umožňuje vývoj nových algoritmů.

Java-ML

Je Java API s kolekcí algoritmů pro strojové učení implementovaných v Javě. Poskytuje pouze standardní rozhraní pro algoritmy.

4.3 Nástroj WEKA

Celá tato kapitola vychází především z informací dostupných v manuálu [1]. Weka – Waikato Environment for Knowledge Analysis – je populární sada softwaru, která obsahuje kolekce algoritmů pro strojové učení a datamining. Zahrnuje v sobě nástroje pro předzpracování dat, klasifikaci, shlukování, asociační pravidla a vizualizaci. Je napsaná v programovacím jazyce Java a byla vyvinuta na univerzitě Waikato na Novém Zélandu [15]. Weka je bezplatný software a je dostupný pod GNU[3]. Weka sama o sobě po stažení a nainstalování obsahuje aplikaci, která umožňuje provádět všechny výše zmíněné možnosti. Avšak pro běžného uživatele, který nemá s tímto nástrojem žádné zkušenost a nemá nastudovánu dokumentaci Weky, může být práce s touto aplikací poněkud problematická. Jelikož je možné importovat a využívat dostupné API Weky ve vlastní aplikaci, může být vytvořena výsledná aplikace, která bude uživatelsky přívětivá a uživatel tak nemusí znát podrobnosti a detaily, jak používat tento nástroj.

[3]GNU - General Public License.

4.3.1 Formát ARFF

Tato kapitola vychází z informací z wiki stránek Weky [16]. Soubor s příponout `.arff` neboli Attribute-Relation File Format je textový ASCII soubor, který popisuje seznam instancí sdílejících sadu atributů. Soubor musí striktně dodržovat určitou syntaxi a definice souboru se skládá ze dvou odlišných částí - hlavičky a datové části.

Na následující ukázce kódu je uveden příklad, jak taková hlavička může vypadat. V hlavičce je definován název relace a seznam jednotlivých atributů. Nejprve je zde uveden komentář, dále výstižný název relace a potom pět atributů. Význam jednotlivých prvků je vysvětlen dále.

```
% komentar

@RELATION test-data

@ATTRIBUTE atribut_1 NUMERIC
@ATTRIBUTE atribut_2 {t}
@ATTRIBUTE atribut_3 {'\'(-inf-1.5]\'','\'(1.5-7.5]\''}
@ATTRIBUTE atribut:4 {TRUE, FALSE}
@ATTRIBUTE atribut_5 class
```

Komentáře

Komentáře se v souborech typu `.arff` píší obvyklým způsobem. Každý komentář začíná uvozovací značkou `%` a pokud má být komentář na více řádcích, rovněž i znak procenta musí být na každém řádku.

Relation

Definice relace musí být deklarována na prvním řádku souboru a udává pouze název zdrojových dat. Název nesmí obsahovat speciální znaky (`{`, `}`, `'` nebo `%`) a pokud obsahuje název mezery, musí být název obklíčen příslušnými uvozovkami. Atribut `relation` žádným způsobem neovlivňuje pozdější učení modelu.

Attribute

Jednotlivé atributy tvoří v souboru posloupnost a udávají pozici konkrétního sloupce v datové části. Například pokud máme definovaný atribut a jeho pozice je na třetím řádku, Weka bude pro tento atribut očekávat hodnoty na třetí oddělující čárce v datové sekci. Každý atribut musí obsahovat svůj název a typ. Název atributu rovněž nesmí obsahovat speciální znaky.

Datový typ může nabývat těchto čtyř podporovaných typů:

Numeric

- Numerický atribut může být jak celé, tak i reálné číslo.

Nominal

- Hodnoty nominálního atributu jsou definovány výčtem jednotlivých prvků množiny.
- Hodnoty jsou uvozeny ve složených závorkách. Jestliže chceme v hodnotách uvádět speciální znaky, je potřeba znak opatřit zpětným lomítkem, případně pokud by měl název obsahovat mezery, je nutné použít uvozovky pro daný prvek.
- Speciálním typem nominálního atributu je atribut typu `class`, který je stěžejní pro predikční úlohy.

String

- Atribut typu string umožňuje vytvářet libovolné textové hodnoty. Tento typ se velice často používá v aplikacích textového dolování.
- Weka nabízí vhodné filtry pro manipulaci s textovými řetězci (např. `StringToWordVectorFilter`).

Date

- Tento typ určuje datum. Je nutné uvést jméno atributu, pro který platí stejná omezení týkající se názvu.
- Volitelně je možné uvést parametr pro formát data a času. Tento parametr pak určuje, jak se budou výsledné hodnoty tisknout.
- Hodnoty musejí být v datové části reprezentovány jako stringové hodnoty.

V ukázce kódu zmíněného výše, je uvedeno pět atributů. První atribut může nabývat číselných hodnot a má typ `numeric`, druhý, třetí a čtvrtý atribut jsou nominálního typu a tyto atributy mohou nabývat předem určených hodnot. Poslední atribut má speciální typ `class` sloužící pro predikční úlohy.

Druhá část souboru `.arff` je datová část. Datová část musí začínat speciálním tokenem `@data`. Jetnotlivé hodnoty se potom oddělují čárkami a udávají jednotlivé sloupce. Pokud u atributu není definovaná hodnota, je zde otazník. Otazník hraje rověž důležitou úlohu u atributu typu `class`, kde indikuje, že se jedná o nová data, která chceme klasifikovat a zařadit do konkrétních tříd.

Na ukázce níže je možné vidět, jak vypadá datová část:

```
@data
1,t,'\'(-inf-1.5]\'',TRUE,?
2,?,'\'(-inf-1.5]\'',TRUE,?
3,t,'\'(1.5-7.5]\'',FALSE,?
4,?,'\'(-inf-1.5]\'',FALSE,?
5,?,'\'(1.5-7.5]\'',FALSE,?
```

Sparse ARFF

Sparse, případně jej můžeme nazvat řídký ARFF soubor, je velmi podobný klasickému ARFF soboru. Jediné, v čem se liší, je datová část. Místo toho, aby datová část obsahovala všechna data, Sparse soubor obsahuje pouze nenulové hodnoty a jejich indexy a data musejí být ve složených závorkách. Vynechané hodnoty jsou však pouze nuly, nikoliv chybějící hodnoty (reprezentovány otazníkem).

Rozdíl mezi ARFF a Sparse ARFF je na následující ukázce:

```
% Bezny arff soubor
@data
0, X, 0, Y, "class A"

% Sparse arff soubor
@data
{1 X, 3 Y, 4 "class A"}
```

4.3.2 Dataset

Dataset (množina datových položek) je základní koncept strojového učení. Dataset odpovídá zhruba dvojrozměrné tabulce nebo databázové tabulce. Ve Wece je k tomu využívána naimplementovaná třída `weka.core.Instances`. Každá instance se skládá z množiny atributů a datové části. Reprezentací instancí je právě soubor ARFF popsaný v části 4.3.1. Balík `weka.core` nabízí mimo jiné užitečné funkce, které slouží pro zpracování instancí a import ze zdrojových dat.

4.3.3 weka.filters

Balík `weka.filters` se skládá z mnoha tříd umožňující transformaci daného datasetu. Mezi nejběžnější operace patří odstranění či přidávání atributů, převzorkování datové sady a mnoho dalších. Tento balík má vysokou podporu pro předzpracování dat, které je velice důležité při strojovém učení. Balík `weka.filters` je organizován do dvou částí tzv. `supervised` a `unsupervised` filtrování. Stejné funkce v jednotlivých částech pak mohou mít jiné chování.

weka.filters.unsupervised

U tohoto případu je nutné, aby byl zadán třídní atribut informace o něm. Ve výchozím nastavení je třídní atribut nastaven vždy jako poslední. Pokud bychom chtěli jiný atribut jako třídní, je nutné jej nastavit pomocí parametru *-c*.

Pro atributy jsou typickými zástupci metody `Discretize`, `NominalToBinary` či `Resample`.

Discretize

- V prvním případě se jedná o diskretizování numerických atributů na nominální. Některé klasifikátory totiž umějí pracovat pouze s nominálními atributy.
- Diskretizování může výrazně redukovat čas učení. U supervised filtrování je diskretizování hodnot založeno na informaci o třídě.

– U unsupervised je diskretizování hodnot založeno na rozdělení do košů.

NominalToBinary

– Využívá se pro zakódování nominálních atributů do binárních. Většinou dvouhodnotových binárních atributů, které se použijí k transformaci dat na čistě numerickou reprezentaci.

Resample

– Pro instance je vhodné použít například metodu Resample, která vytvoří stratifikovaný dílčí vzorek dat a zachovává při tom přibližně stejné rozdělení třídních instancí v daném vzorku.

weka.filters.supervised

Třídní atributy by zde neměly být přiřazeny. Mezi běžné metody, které se používají patří např. `StringToWordVector`, `Obfuscate` či `Remove`.

StringToWordVector

– Využívá se především u úloh zabývajících se dolováním z textu. Transformuje daný řetězec na vektor slov.

– Vytvoří například jeden atribut pro každé slovo, kde kóduje přítomnost nebo počet výskytů.

Obfuscate

– Obfuscate je metoda užitečná pro přejmenování datové sady, názvů atributů a nominálních hodnot.

Remove

– Je metoda určena pro explicitní odstranění atributů z daného datasetu. Odstranění probíhá zadáním příslušného intervalu (pokud chceme odstraňovat více atributů) nebo jen indexem. Musíme brát však v úvahu, že indexování je od nuly. Avšak pokud chceme odstranit první atribut, je nutné přičíst jedničku.

4.3.4 weka.associations

Balík `weka.associations` obsahuje kolekci algoritmů pro dolování asociačních pravidel a frekventovaných množin. Obsahuje abstraktní třídu `AbstractAssociator`, která slouží pro učení. Dále obsahuje implementaci algoritmů `Apriori` a `FP Growth` včetně získání frekventovaných množin a metod pro práci s jednotlivými prvky a získání jejich informací.

4.3.5 weka.classifiers

Jakýkoliv učící algoritmus je ve Wece odvozen od abstraktní třídy `weka.classifiers.AbstractClassifier` a je implementován ve `weka.classifiers.Classifier`. Sestavení základního klasifikátoru je poměrně jednoduché. Nejprve je volána metoda generující klasifikační model z trénovacího datasetu (`buildClassifier`), následně další metoda, která vyhodnocuje model na doposud neznámých testovacích datech (`classifyInstance`) nebo

generuje rozdělení pravděpodobnosti pro všechny třídy (`distributionForInstance`). Klasifikační model je libovolné mapování všech atributů z datasetu do jednoho atributu `class`.

Klasifikátory jsou jádrem Weky. Proto existuje mnoho parametrů, které jsou společné pro všechny klasifikátory.

Zde uvedu jen ty nejdůležitější nastavovací parametry:

`-t` Specifikuje soubor pro trénování modelu (ARFF soubor).

`-T` Specifikuje soubor pro testování (ARFF soubor). Pokud chybí, proběhne křížová validace.

`-x` Určuje počet částí (foldů) pro křížovou validaci. Provede se pouze v případě, že není zadán parametr `-T`.

`-c` Nastavuje třídní atribut.

`-l` Načte již dříve vytvořený a uložený model pro otestování nových dat. Nová data by měla mít stejnou strukturu a pořadí atributů, podle kterých byl model naučen.

`-i` Zobrazí podrobnější výpisy statistik.

`-o` Tento parametr převede výstup do čitelné podoby pro člověka.

Kapitola 5

Popis implementace výsledné aplikace

Primárním cílem této práce je vytvořit aplikaci, která umožňuje vhodným a jednoduchým způsobem získat potřebné informace o produktech, které zákazníci kupují. Tyto informace mohou posloužit k vhodnějšímu nabízení produktů a oslovování zákazníků. Aplikace vychází z úlohy č. 1 popsané v kapitole 3.3. Byly zvoleny oba algoritmy pro dolování asociačních pravidel a to algoritmus Apriori a FP Growth. Aplikace se skládá z jednotlivých záložek, kdy každá z nich umožňuje specifickou práci. Aplikace také umožňuje zpracovat již vyexportované vstupní soubory s daty zákazníků a produktů a umí je zpracovat do požadovaného formátu ARFF pro jednotlivé úlohy. Výsledky dolování zobrazuje do ineraktivní tabulky, ve které je možné vyhledávat či filtrovat. Navíc byla aplikace rozšířena o další záložku predikce zákazníka vycházející z úlohy č. 2 popsané v kapitole 3.4, ve které je možné nastavit seznam produktů a parametrů a dotázat se na konkrétní produkt. Byla vybrána predikční metoda Naive Bayes, protože byla jednou z metod, které dávaly v rámci experimentu nejlepší přesnost. Aplikace vytvoří model (případně použije již naučený) a na základě vstupních dat určí třídu, která určí, zda si zákazník produkt koupí nebo ne.

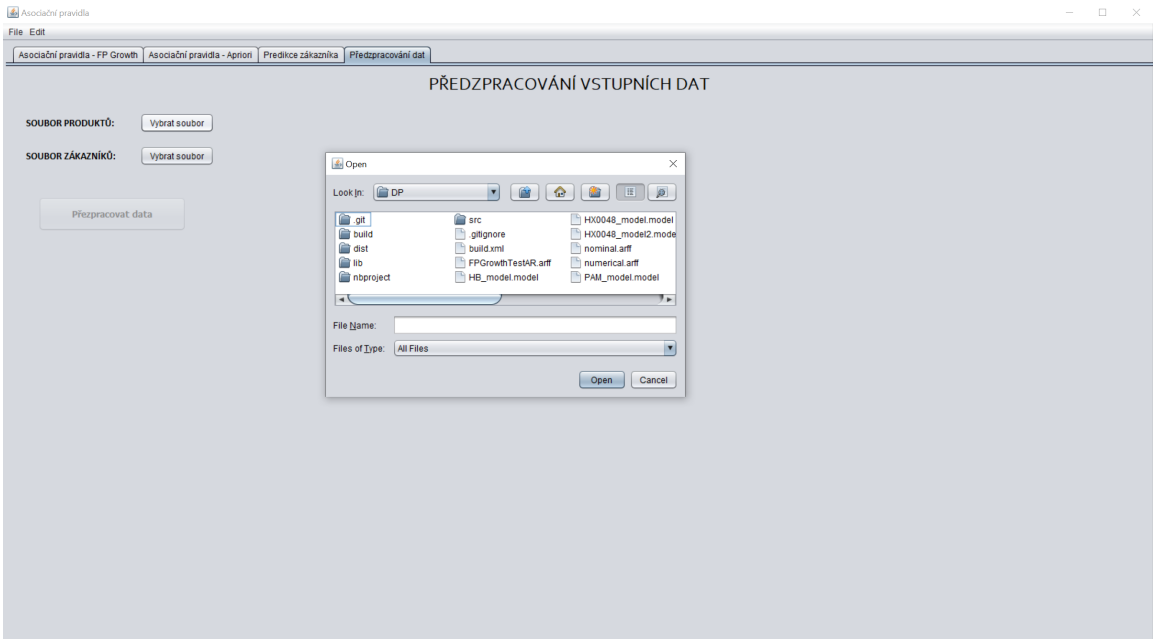
V první části je popsáno předzpracování vstupních dat. Dále je popsáno, jak bylo implementováno dolování asociačních pravidel a nakonec predikce zákazníka. V příloze se nachází diagram tříd B.1 aplikace.

5.1 Předzpracování dat

Vstupní data pro předzpracování jsou ve formátu `CSV`. S tímto formátem Weka dokáže bez problémů pracovat. Avšak z důvodu, že se data musejí stejně upravovat do výsledné podoby pro predikční úlohy, byly v rámci předzpracování vytvořeny dva nové soubory ve formátu ARFF. Jedná se o soubor `nominal.arff` a `numerical.arff`. Soubor `nominal.arff` je potom použit pro úlohu dolování asociačních pravidel a soubor `numerical.arff` pro predikci zákazníka. Oba tyto soubory jsou vytvořeny ze souborů `zakaznici.csv` a `produkty.csv`.

V záložce *Předzpracování dat* je možné právě tyto dva soubory vytvořit – obrázek 5.1. V levé části jsou možnosti pro zvolení souborů. Nejprve se zvolí vstupní soubory s daty zákazníků a s daty produktů. Posléze se zpřístupní tlačítko pro spuštění předzpracování. Jelikož jsou soubory `CSV` vyexportovány z firemní databáze, je nutné, aby měly striktně dodržen vstupní formát dat, protože jej v tomto pořadí aplikace očekává a zpracovává. U produktů je nutné, aby první element obsahoval zkratky produktů. U zákazníků je pořadí

elementů následovné: firma, stav zákazníka, počet pracovníků, důležitost a seznam aplikací.



Obrázek 5.1: Předzpracování dat + dialogové okno pro výběr souboru

Samotné zpracování dat probíhá následovně. Projde se soubor s produkty. Pro předzpracování slouží třída `filePreprocessing`. V metodě, která zpracovává data, je objekt typu `TreeMap<String, Integer>`, do kterého se ukládají zkratky jednotlivých produktů a jejich hodnota. `TreeMap` navíc udržuje celý seznam setříděný. Takže všechny zkratky produktů jsou seřazeny dle abecedy. Dále se prochází soubor se zákazníky a plní se mapa jednotlivými daty. Ve sloupci se zakoupenými produkty se kontroluje, zda zkratky opravdu odpovídají zkratkám načtených ze souboru produktů. Pokud by se zkratka nenašla, přidá se nový produkt do seznamu. Tento seznam uchovává jedničky nebo nuly podle toho, zda má zákazník produkt koupený či nikoliv. Rozdíl u nově vytvořených soborů `nominal.arff` a `numerical.arff` je v tom, že soubor `nominal.arff` obsahuje nominální hodnotu `t` pro zakoupený produkt, v ostatních případech je hodnota nedefinovaná znakem `?`. Soubor `numerical.arff` obsahuje nuly a jedničky.

Některé produkty obsahují v názvu znak `*`. Např. `HB*`. Produkt HB znamená, že se jedná o bankovní produkt. Hvězdička však udává, že zákazník má koupených více bankovních modulů. Proto byly nastaveny i všechny ostatní produkty začínající HB rovněž na jedničku (tedy zakoupeno).

V posledním kroku předzpracování je použito diskretizování hodnot atributů počtu zakoupených produktů a počtu pracovníků. Je použita metoda `Discretize` [4] s následujícími parametry: `"-O", "-R", "2"`. Parametr `-O` určuje diskretizaci atributů se zhruba stejnou šířkou jednotlivých košů. Parametr `-R` určuje jen konkrétní atribut, který se má diskretizovat. V tomto případě se jedná o atribut počtu produktů. Pro počet pracovníků byl použit parametr `-F`, který místo rozdělení podle stejné šířky používá rozdělení dle frekvence výskytů.

5.2 Asociační pravidla a frekventované množiny

Aplikace umožňuje dolování asociačních pravidel a frekventovaných množin dvojím způsobem. První způsob využívá algoritmus FP Growth. Druhý způsob dolování je pomocí algoritmu Apriori. Algoritmy jsou popsány v části 2.3.2.

Jelikož oba způsoby dolování využívají obdobné nastavení či parametry, byla navržena abstraktní třída `AssotiationRulesCtr` a k ní dvě třídy `FPGrowthCtrl` a `AprioriCtrl`, které tuto třídu rozšiřují a implementují v sobě metody `start()` pro samotné dolování a `processOutput()` pro zpracování výstupu algoritmu do čitelné podoby pro uživatele.

5.2.1 Asociační pravidla - FP Growth

V rámci aplikace byl využit algoritmus FP Growth. Je vhodný pro účely dolování a získání zajímavých asociací z toho důvodu, že u něj lze navíc nastavit parametr, který produkt či produkty by měly být ve výsledných asociačních pravidlech zahrnuty. Po spuštění aplikace se uživateli zobrazí jako první právě tato záložka.

V horní části se nachází výběr parametrů a filtr dat. V možnostech lze nastavit počet výsledných pravidel a hodnoty podpory a spolehlivosti. Filtr dat slouží k lepšímu zaměření na vstupní soubor s daty, ze kterých budou pravidla dolována. V nastavení filtru dat je možnost (checkbox) pro spuštění nad všemi daty. Pokud není nastaven, jsou zpřístupněny selecty na výběr počtu pracovníků a počtu zakoupených produktů. Jestliže nejsou vybrána všechna data, po spuštění se pomocí metody `Remove` 4.3.3 vyfiltrují pouze ta vstupní data, která obsahují právě tyto dva zvolené parametry. Dolování tedy probíhá na menším počtu dat.

Další parametr pro nastavení je výběr produktů, které by měly být obsaženy ve výsledných asociačních pravidlech. Zvolený produkt se přidá do seznamu a je uživateli zobrazen. Pokud by uživatel chybně zvolil produkt, aplikace mu umožňuje, aby mohl poslední produkt ze seznamu odstranit.

Poslední možnost, která se v této záložce nachází, je zvolení jiného vstupního souboru, např. s novými daty. Po kliknutí na toto tlačítko se uživateli zobrazí možnost pro výběr soboru. Ve výchozím nastavení je, že se bere právě soubor `nominal.arff`. Pokud neexistuje, případně je ve špatném formátu, nelze pochopitelně provádět žádné dolování.

Po spuštění pro vybrané parametry a nastavené hodnoty počtu pravidel, podpory a spolehlivosti, se vytvoří objekt `FPGrowthCtrl` a zavolá se metoda `start()` (sekvenční diagram metody `start()`, která provádí dolování, je možné vidět v příloze C.1). V této metodě je načten vstupní soubor. Dále je použit filtr `Remove` 4.3.3 a jsou odstraněny nepotřebné atributy. Soubor pro dolování obsahuje tedy jen čistě nominální data produktů. Před tím jsou však samozřejmě ještě vybrány instance podle zadaných filtrů. Aplikace zároveň vytvoří ještě objekt `ArffSaver`, který umožní vytvoření nového souboru `FPGrowthTestAR.arff`, ze kterého bude probíhat dolování. Následně se vytvoří objekt `FPGrowth()` a pomocí metody `buildAssociations(data)` získáme výsledná pravidla.

Parametry, se kterými byl `FPGrowth()` spuštěn, jsou následující:

- `-S` Najde všechna pravidla, která splňují dolní hranici minimální podpory a minimální metrické omezení.
- `-C` Nastaví hodnotu spolehlivost.
- `-M` Nastaví hodnotu podpory.

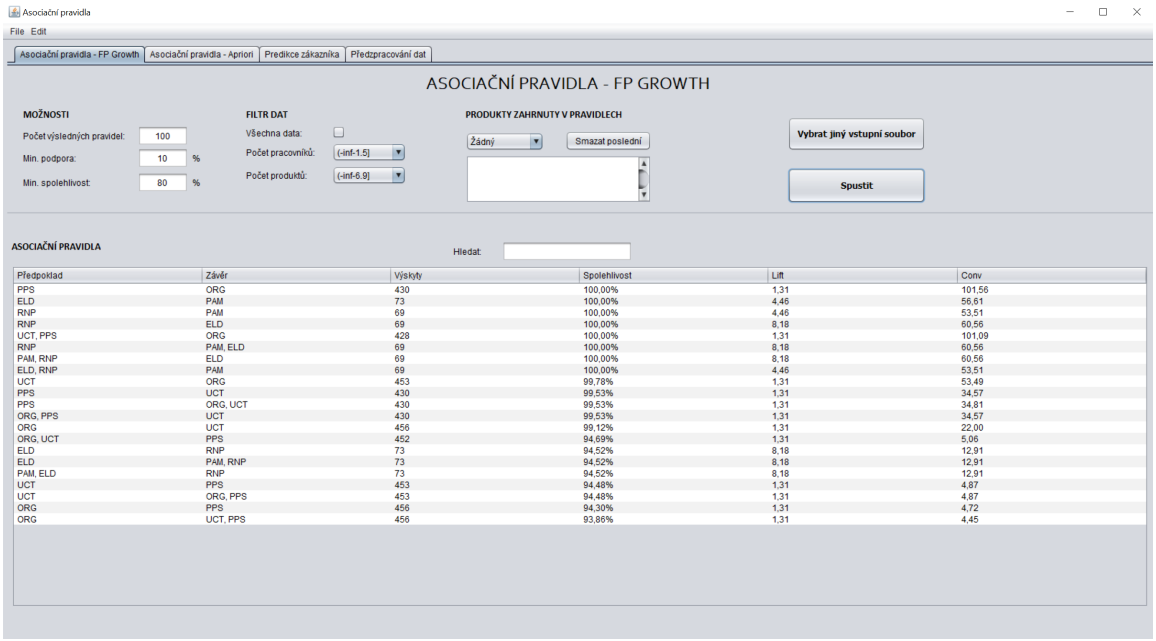
`-N` Nastaví počet výsledných pravidel. V případě, že je tento parametr vyšší než počet instancí v soboru, použije se počet instancí.

`-rules` Nastaví seznam produktů, které by měly být zahrnuty ve výsledných pravidlech.

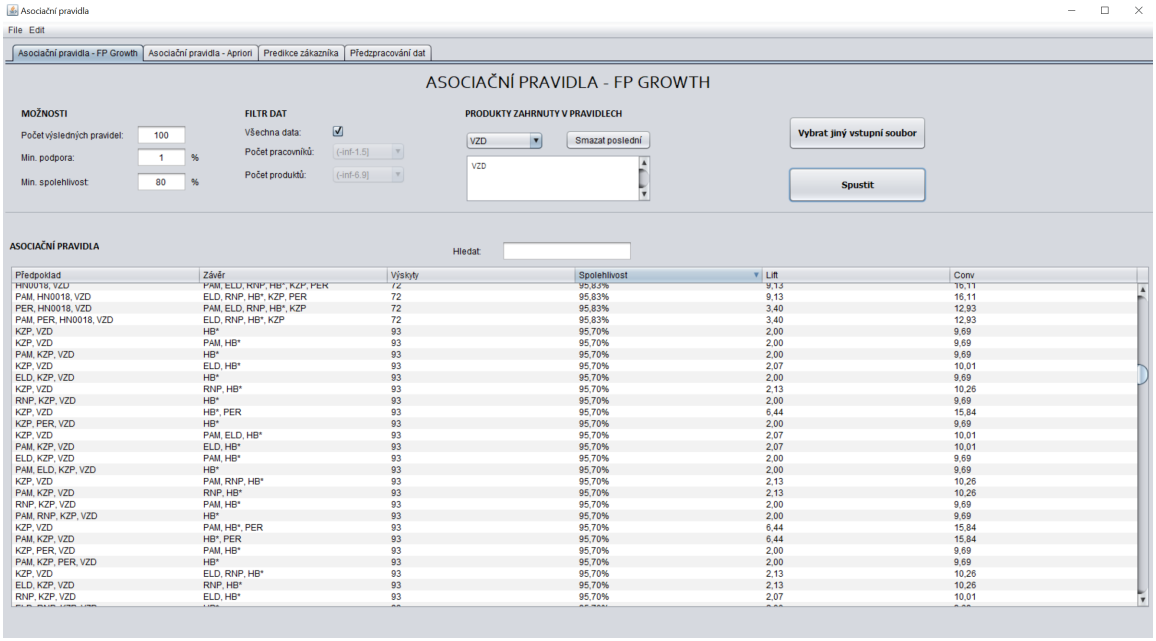
Posléze je výstup načten do `HashMap` a výstup je vhodně transformován pomocí metody `processOutput()` do podoby pro zobrazení v tabulce.

Výstupem je tedy výsledná tabulka. Tabulka obsahuje v prvním sloupci levou stranu asociačního pravidla a v druhém sloupci pravou stranu asociačního pravidla. V dalších sloupcích jsou hodnoty pro jednotlivá pravidla. Jedná se o počet výskytů, celkovou spolehlivost, hodnotu Lift a Conv. Výsledná tabulka je interaktivní a umožňuje řazení podle jednotlivých sloupců, případně je nad tabulkou vstupní pole pro zadání hledaného řetězce. Pro filtrování pomocí jednotlivých sloupců je využit `TableRowSorter` [8]. Pro hledaný zadaný výraz je kromě samotného `TableRowSorter` použit ještě `RowFilter.regexFilter(querry)` [7].

Pro hledání dalších asociací je vhodné snížit hodnotu podpory. Na obrázku 5.2 je možné vidět výsledek po spuštění s použitím filtrů na zákazníky a produkty. Na následujícím obrázku 5.3 je zobrazen výsledek asociačních pravidel, které v sobě navíc zahrnují produkt `VZD`. Aby bylo možné získat tato pravidla, bylo nutné nastavit podporu na 1%.



Obrázek 5.2: Asociační pravidla - FP Growth (použit filtr dat)



Obrázek 5.3: Asociační pravidla - FP Growth (nad všemi daty + zahrnut produkt VZD)

5.2.2 Asociační pravidla - Apriori

Ve druhé záložce je implementováno získání asociačních pravidel za pomocí algoritmu Apriori. V horní části se nachází obdobné nastavení jako u předešlé záložky s algoritmem FP Growth. Chybí zde však nastavení, aby pravidla obsahovala konkrétní produkty. Třídění a také filtry počtu pracovníků a počtu produktů fungují stejným způsobem. Rovněž i vybrání jiného vstupního souboru.

Co tato sekce nabízí navíc je zobrazení frekventovaných množin a dále možnost získat další asociace. Po nastavení parametrů počtu pravidel, podpory a spolehlivosti (případně filtrů) a stisknutí tlačítka *Spustit*, se vytvoří objekt `AprioriCtrl` a je volána funkce `start()`. Získají se instance ze zvoleného vstupního souboru, jsou odstraněny nepotřebné atributy pomocí `Remove` a je vytvořen soubor pomocí `ArffSaver` s názvem `testAR.arff`. Dále je vytvořen objekt `Apriori` a nad vytvořenými daty je zavolána funkce `buildAssociations()`.

Apriori byl spušten s následujícími parametry:

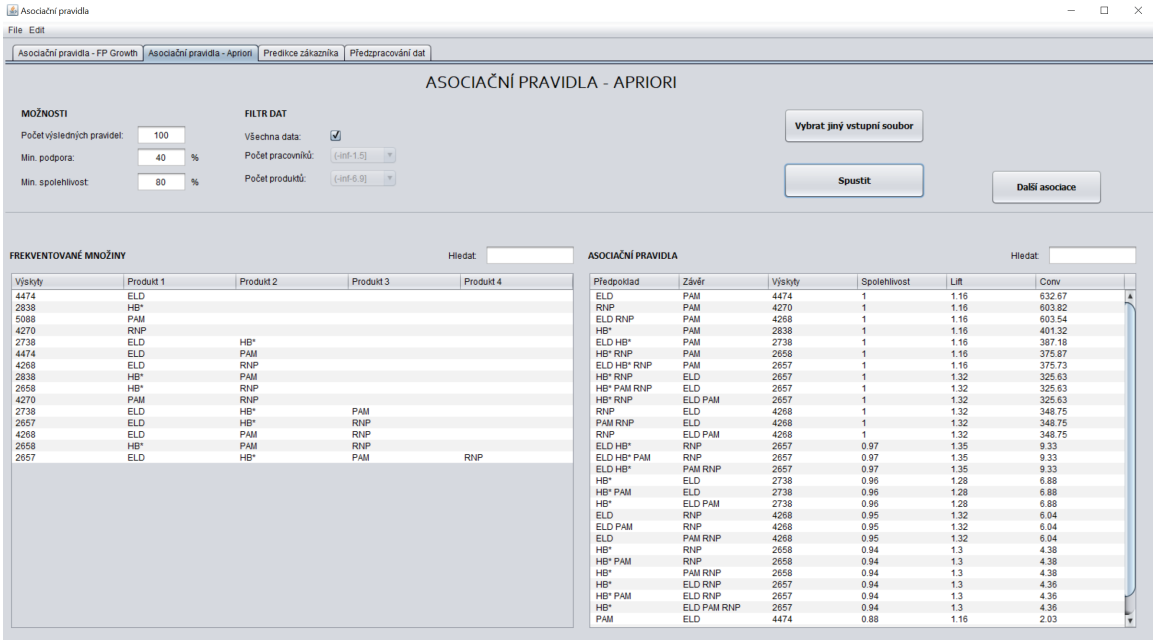
`-C` Nastaví hodnotu spolehlivost.

`-M` Nastaví hodnotu podpory.

`-N` Nastaví počet výsledných pravidel. V případě, že je tento parametr vyšší než počet instancí v soboru, použije se počet instancí.

`-I` Výstup algoritmu bude obsahovat i frekventované množiny.

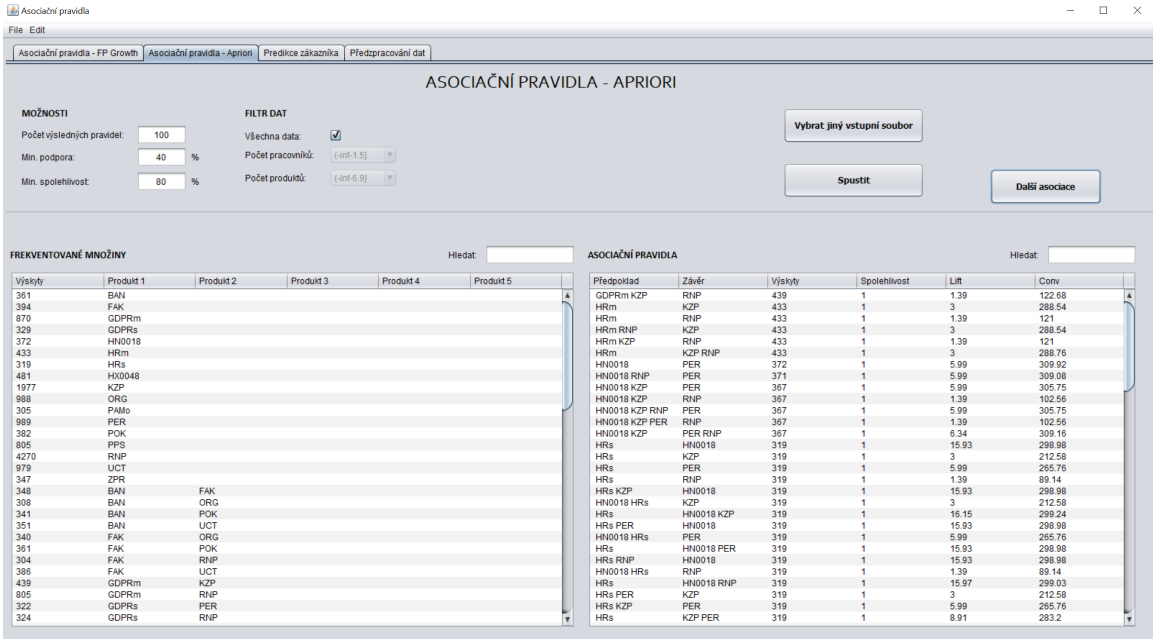
Jakmile jsou výsledná pravidla nalezena, jsou rovněž zpracována do formy pro zobrazení v tabulce pro výsledná pravidla a pro tabulku s frekventovanými množinami. Obě tabulky umožňují opět řazení podle jednotlivých sloupců nebo vyhledávání dle zadaného textu. Výsledek je možné vidět na obrázku 5.4.



Obrázek 5.4: Asociační pravidla - Apriori

Dale je zpřístupněno tlačítko *Další asociace*. Toto tlačítko funguje následujícím způsobem. Spustí se nové dolování, avšak vezmou se hodnoty všech jednoprvkových frekventovaných množin. Tyto produkty se ze vstupního souboru odstraní a dolování probíhá na zbytku dat. Princip dolování dalších asociačních pravidel ukazuje diagram komunikace v příloze D.1.

Dále v rámci experimentů musela být nastavena podpora na velice nízkou hodnotu – 1%, aby mohla být tato pravidla získána. Pokud se klikne na tlačítko znovu, další produkty se přidají do seznamu a znovu se odstraní. Celý proces dolování je možné opakovat. Výsledky dalších nalezených asociací je možné vidět na obrázku 5.5.



Obrázek 5.5: Asociační pravidla - Apriori (Další asociace)

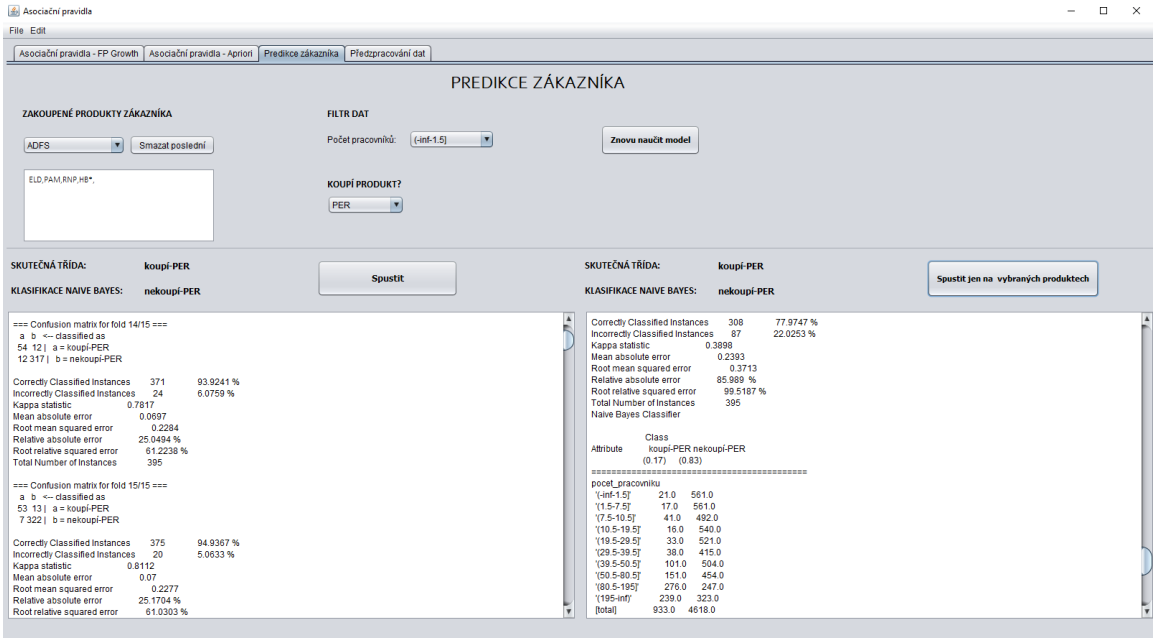
5.3 Predikce zákazníka

Poslední záložkou je *Predikce zákazníka*, která umožňuje zadat parametry zákazníka související s produkty. Využívá se zde vytvoření nebo načtení klasifikačního modelu Naive Bayes.

V levé části aplikace se nachází výběr produktů, které má potenciální zákazník zakoupené. Produkty se přidávají do seznamu. Další nastavení je počet pracovníků a nejdůležitější část je výběr produktu, u kterého chceme zjistit, zda by si jej s takovouto vybranou kombinací zákazník zakoupil. Aplikace je rozdělena na dvě části (obrázek 5.6). V levé části se nachází dolování nad všemi produkty. V pravé části naopak proběhne odstranění všech produktů a zůstanou jen ty, které jsou vybrány uživatelem. Po stisknutí tlačítka *Spustit* se vytvoří objekt typu `NaiveBayesCtr` a spustí se metoda `start()`. Nejprve se zkontroluje, zda už pro tento produkt není model vytvořen. Pokud není, zavolá se metoda `learnModel()` a proběhne učení nového modelu. Sekvenční diagram metody `learnModel()`, která provádí dolování, je možné vidět v příloze C.3.

Učení probíhá tak, že se načte vstupní soubor a procházejí se jednotlivé instance. Uloží se seznam atributů, dále se získá index produktu, který zkoumáme a vytvoří se nový atribut typu `class`, který nabývá hodnot *Koupí/Nekoupí produkt X*. Dále se vytvoří nový soubor `unknown.arff`, který obsahuje stejné vstupní atributy jako ze vstupních dat a dále řádek s novými neznámými daty, které jsou vytvořeny na základě parametrů, které zadal uživatel.

Vytvoří se objekt `NaiveBayes` a probíhá učení na vstupních datech pomocí stratifikace. Náhodně se zamíchají jednotlivé instance a data jsou rozdělena na testovací a učící. Učení zahrnuje všechny atributy produktů. Dále probíhá v 15-ti cyklech učení modelu a v každé iteraci je zaznamenána matice záměn klasifikátoru. Po ukončení učení je model uložen pomocí `SerializationHelper`.



Obrázek 5.6: Predikce zákazníka - Naive Bayes

Následně se model použije na neznámá data `unknown.arff`. Na základě toho jsou neznámá data klasifikátorem zařazena do nové třídy. Zařazení klasifikátorem je zobrazeno nad textovým oknem a je zde zobrazena skutečná třída a také predikovaná třída. Pokud proběhlo učení modelu, součástí výstupu je také výpis 15-ti iterací s maticemi záměn a chybami. Také je zde výpis statistik pro jednotlivé atributy. Například počet instancí v jednotlivých třídách podle počtu pracovníků, případně pro každý atribut jeho váhy a další hodnoty.

Dále jsem ještě provedl experiment, jak by se choval model, pokud by se učil jen na datech s atributy produktů, které zadal uživatel. Uživatel tedy vybere například 5 produktů a dotáže se na konkrétní produkt. Ze vstupních dat se zjistí indexy právě těchto produktů a ostatní produkty se odstraní a proběhne učení nového modelu. Když jsem porovnával výsledky přesnosti klasifikace v jednotlivých iteracích, došlo ke zhoršení celkové přesnosti modelu. Výsledky jsou popsány v následující kapitole 6.

Kapitola 6

Ověření funkčnosti

Výsledky dolování asociačních pravidel jednotlivých metod byly ověřovány vůči výsledkům získaných v programu RapidMiner.

Ve druhé záložce aplikace, kde je dolování asociačních pravidel pomocí algoritmu Apriori, se nacházejí v levé části nalezené frekventované množiny (obrázek 6.1). Ve výstupu aplikace je místo hodnoty podpory hodnota počtu výskytů. Hodnota podpory byla nastavena na 40% a hodnota spolehlivosti na 80%. Stejně jako v programu RapidMiner v úloze č. 1 popsané v kapitole 3.3. V tabulce A je možné vidět frekventované množiny získané v RapidMineru. V RapidMineru je pro dolování použit algoritmus FP Growth. Jestliže porovnáme výsledky aplikace s výsledky z RapidMineru, frekventované množiny jsou shodné, rozdíl je jen v abecedním seřazení produktů.

FREKVENTOVANÉ MNOŽINY Hledat:

Výskyty	Produkt 1	Produkt 2	Produkt 3	Produkt 4
4474	ELD			
2838	HB*			
5088	PAM			
4270	RNP			
2738	ELD	HB*		
4474	ELD	PAM		
4268	ELD	RNP		
2838	HB*	PAM		
2658	HB*	RNP		
4270	PAM	RNP		
2738	ELD	HB*	PAM	
2657	ELD	HB*	RNP	
4268	ELD	PAM	RNP	
2658	HB*	PAM	RNP	
2657	ELD	HB*	PAM	RNP

Obrázek 6.1: Apriori - frekventované množiny

Vezmeme-li například pravidlo $ELD, RNP \Rightarrow PAM$ z tabulky 3.1. Hodnota spolehlivosti je 100%, dále v RapidMineru je možné dohledat hodnotu $Lift = 1.019$, $Conviction = \infty$. Aplikace dává pro toto pravidlo a použití algoritmu Apriori hodnotu spolehlivosti také 100%, $Lift = 1.16$ a $Conviction = 603.54$. Dále pokud použijeme algoritmus FP Growth v první záložce a nastaví se filtr, aby pravidla obsahovala konkrétní produkty, najdeme například pravidlo $PAM, UCT \Rightarrow FAK$. Pro toto pravidlo byly hodnoty v RapidMineru následující: spolehlivost 79.9%, $Lift = 11.56$ a $Conviction = 4.981$. Aplikace dala pro toto pravidlo hodnoty spolehlivost 71.52%, $Lift = 10.76$ a $Conviction = 3.25$. Hodnoty se liší řádově v jednotkách. Faktem však je, že oba algoritmy použité v aplikaci naleznou stejná

asociační pravidla jako RapidMiner. Navíc se v aplikaci, za použití různých filtrů, mohou nalézt pravidla pouze pro vybrané skupiny zákazníků.

Predikci zákazníka ukážu na následujícím příkladu, kde provedu zhodnocení výsledků. Ze vstupních dat jsem vybral náhodně zákazníka, který má zakoupeny tyto produkty: $ELD, PAM, RNP, HB*, PER$. Budeme chtít zjistit, zda si koupí například produkt PER. Nejprve model naučíme na datech, která zahrnují všechny produkty (matice záměn RapidMiner 6.2 a výsledky aplikace 6.4) a posléze vyfiltrujeme jen vybrané produkty (matice záměn RapidMiner 6.3 a výsledky aplikace 6.5).

accuracy: 88.89%

	true false	true true	class precision
pred. false	419	43	90.69%
pred. true	20	85	80.95%
class recall	95.44%	66.41%	

Obrázek 6.2: Matice záměn metody Naive Bayes zahrnující všechny produkty (RapidMiner)

accuracy: 87.83%

	true false	true true	class precision
pred. false	407	37	91.67%
pred. true	32	91	73.98%
class recall	92.71%	71.09%	

Obrázek 6.3: Matice záměn metody Naive Bayes zahrnující konkrétní produkty (RapidMiner)

SKUTEČNÁ TŘÍDA: **koupí-PER**

KLASIFIKACE NAIVE BAYES: **nekoupí-PER**

Correctly Classified Instances	355	89.8734 %
Incorrectly Classified Instances	40	10.1266 %
Kappa statistic	0.6225	
Mean absolute error	0.1044	
Root mean squared error	0.297	
Relative absolute error	37.5257 %	
Root relative squared error	79.619 %	
Total Number of Instances	395	

```
=== Confusion matrix for fold 8/15 ===
  a  b  <-- classified as
 51  15 |  a = koupí-PER
 10 319 |  b = nekoupí-PER
```

Obrázek 6.4: Všechny produkty

SKUTEČNÁ TŘÍDA: **koupí-PER**

KLASIFIKACE NAIVE BAYES: **nekoupí-PER**

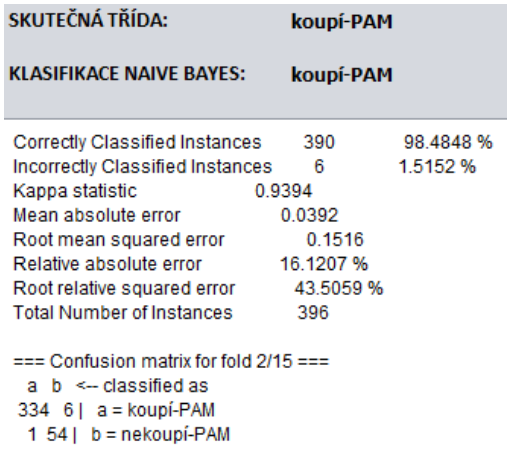
Correctly Classified Instances	325	82.2785 %
Incorrectly Classified Instances	70	17.7215 %
Kappa statistic	0.4875	
Mean absolute error	0.2104	
Root mean squared error	0.3392	
Relative absolute error	75.6206 %	
Root relative squared error	90.9312 %	
Total Number of Instances	395	

```
=== Confusion matrix for fold 5/15 ===
  a  b  <-- classified as
 51  15 |  a = koupí-PER
 63 266 |  b = nekoupí-PER
```

Obrázek 6.5: Vybrané produkty

Přesnost klasifikátorů naučených na datech zahrnující všechny produkty je skoro stejná (RapidMiner 88.89%, aplikace 89.87%). Na datech obsahující pouze vybrané produkty je přesnost klasifikátoru v aplikaci horší a dopustil se většího počtu špatného zařazení do správné třídy (RapidMiner 87.83%, aplikace 82.27%). Učení probíhalo na náhodném výběru dat pro testování a pro ověřování. RapidMiner použil větší počet dat než aplikace (cca 500 záznamů v RapidMineru a cca 400 v aplikaci). Nezávisle na použitém přístupu (nad všemi produkty/vybranou množinou) je v aplikaci výsledné zařezení do tříd stejné. Skutečná třída znamená to, že zákazník by si měl tento produkt koupit (v datech se vy-

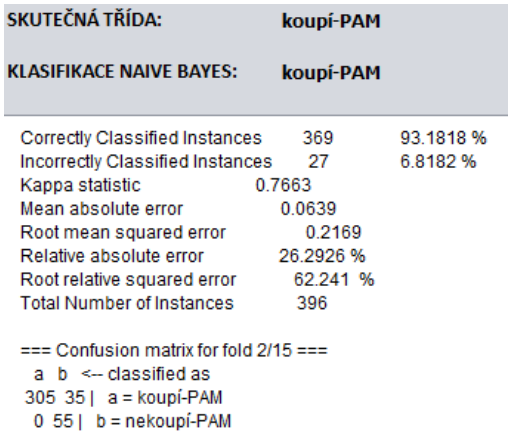
skytli zákazníci s touto kombinací). Klasifikátor však určil, že zákazník produkt nekoupí. V matici záměn je totiž z celkového počtu dat, které byly použity, jen 51 zákazníků, kteří jej opravdu mají s touto kombinací produktů. Podíváme-li se na produkt PAM, o kterém víme, že patří mezi nejvíce četné produkty, klasifikátor určí, že si produkt určitě zákazník koupí (obrázky 6.6 a 6.7).



SKUTEČNÁ TŘÍDA:	koupí-PAM	
KLASIFIKACE NAIVE BAYES:	koupí-PAM	
Correctly Classified Instances	390	98.4848 %
Incorrectly Classified Instances	6	1.5152 %
Kappa statistic	0.9394	
Mean absolute error	0.0392	
Root mean squared error	0.1516	
Relative absolute error	16.1207 %	
Root relative squared error	43.5059 %	
Total Number of Instances	396	

```
=== Confusion matrix for fold 2/15 ===
  a   b   <-- classified as
334   6 |   a = koupí-PAM
  1  54 |   b = nekoupí-PAM
```

Obrázek 6.6: Všechny produkty



SKUTEČNÁ TŘÍDA:	koupí-PAM	
KLASIFIKACE NAIVE BAYES:	koupí-PAM	
Correctly Classified Instances	369	93.1818 %
Incorrectly Classified Instances	27	6.8182 %
Kappa statistic	0.7663	
Mean absolute error	0.0639	
Root mean squared error	0.2169	
Relative absolute error	26.2926 %	
Root relative squared error	62.241 %	
Total Number of Instances	396	

```
=== Confusion matrix for fold 2/15 ===
  a   b   <-- classified as
305  35 |   a = koupí-PAM
  0  55 |   b = nekoupí-PAM
```

Obrázek 6.7: Vybrané produkty

Kapitola 7

Závěr

V této práci jsem se zabýval analýzou firemních dat o vytíženosti jejich služeb s využitím dolování dat. Firma mi poskytla reálná data svých zákazníků a požadavků od nich. Zisk užitečných informací z těchto dat potom může pomoci firmě nejen v procesu rozhodování, ale i v oblasti obchodu. Neexistuje u nich žádný nástroj, který by jim zjistil informace, které nejsou na první pohled v jejich datech vidět. Proto jsem se rozhodl vytvořit potenciálně zajímavé úlohy a posléze vytvořit aplikaci, která by implementovala některou úlohu. Pro experimenty byl použit nástroj RapidMiner.

První úloha se zabývala asociačními pravidly a nabídkou souvisejících služeb. Data byla vhodně upravena pro experiment v rámci předzpracování. Výsledná data zákazníků obsahují sloupce s počtem osobních čísel, důležitostí a stavem. Dále obsahují sloupce se zkratkami všech produktů. U každého produktu je pak známo, zda ho má zákazník koupený či nikoliv. Nalezená frekventovaná množina nejčetnějších produktů obsahovala čtyři produkty, které převyšovaly ve velké míře všechny ostatní produkty. Jejich hodnota podpory je více než 60%. Jedná se o produkty *PAM* – Mzdy, *ELD* – Elektronické podání, *RNP* – Registrace nemocenského pojištění a *HB** – Bankovní moduly. V této úloze byl zkoumán také konkrétní produkt *VZD* – Vzdělávání, který není tak zastoupený. Chtěl jsem ověřit, jak se chová metoda i pro méně četný produkt a jaké další produkty s ním souvisí.

Druhá úloha částěčně vychází z první. Jsou v ní popsány jednotlivé klasifikační metody, které zjišťují, zda si zákazník s určitými parametry koupí daný produkt a s jakou pravděpodobností. Metody jsou ověřovány také pro produkt *VZD* s cílem zjistit, jaké pro něj dávají výsledky. Jelikož všechny metody měly dobrou úspěšnost, zaměřil jsem se na ty metody, které měly nejlepší zařazení do správné třídy. Nejlepší výsledky dávaly metody *Naive Bayes*, *Gradient Boosted tree* a *Deep learning*.

Poslední úloha spadá do oblasti shlukování a pracuje s požadavky od zákazníků. Za pomocí odlehlých hodnot se snažím najít ty zákazníky, kteří se nějakým způsobem vyčleňují od ostatních. Data požadavků od zákazníků jsou použita od roku 2013 a byla rovněž upravena do výsledné podoby. Časový interval byl rozdělen na kvartály jednotlivého roku a u každého zákazníka, který měl požadavek, se pracovalo s počty požadavků v jednotlivých kvartálech a v dané oblasti. Byla zvolena shlukovací metoda *K-means*. Data byla zvolena za poslední rok. Sledoval jsem nejprve všechny požadavky, včetně nulových požadavků ve všech kvartálech. Následně byl použit filtr pro nenulové požadavky. Metoda nalezla několik shluků zákazníků, kteří se liší od ostatních. Většinou se jednalo o zákazníky z oblasti *PAM*, kteří měli buď hodně požadavků (nebo žádné), případně měli vysokou hodnotu atributu celkového počtu požadavků.

Hlavním úkolem aplikace je zjistění asociačních pravidel z dat zákazníků. Aplikace byla implementována v jazyce Java a využil jsem nástroj Weka, který obsahuje kolekce algoritmů pro strojové učení a datamining. Aplikace zpracuje vyexportované `csv` soubory a předzpracuje je do požadovaného formátu pro dolovací úlohy. V aplikaci jsou využity algoritmy Apriori a FP Growth pro dolování asociačních pravidel a frekventovaných množin. U algoritmu FP Growth, který je z hlediska výkonu rychlejší, je navíc možné zadat požadavky na produkty, které by měly být obsaženy ve výsledných pravidlech. U algoritmu Apriori je implementována možnost pro dolování dalších asociací, kdy na úkor snížení hodnoty podpory je možné nacházet další asociace u méněčetných produktů. U obou algoritmů je možné použití filtru dat, stejně tak je možné se zaměřit jen na ty skupiny zákazníků, kteří mají například počet pracovníků (licencí) v konkrétním intervalu. Výsledky jsou zobrazeny v přehledné tabulce, která umožňuje vyhledávání či filtrování. Výsledky algoritmů se shodují s experimenty, které byly prováděny v programu RapidMiner s tím, že aplikace umožňuje více filtrovat vstupní data, od čehož se také odvíjí výsledná pravidla.

Dále byla aplikace rozšířena o možnost predikce chování zákazníka, kdy na základě vybraného seznamu produktů a dotazu na konkrétní produkt, se pomocí klasifikační metody Naive Bayes vytvoří a naučí model a klasifikuje se nová instance. Nová instance reprezentuje data, která model ještě nezná a model by měl určit třídu, zda by si tento produkt zákazník koupil. Jestliže model ještě nebyl vytvořen, provede se učení, data se náhodně rozdělí na testovací a učící. V 15-ti iteracích se model učí a pro každou iteraci je zobrazena matice záměn. Dále zde byl proveden experiment, který podle vybraných produktů prováděl učení na omezeném počtu atributů. Výsledný model měl však horší přesnost.

Výsledná aplikace a její funkčnost byla prezentována a ukázána řediteli Divize Obchod, který se zákazníky nejčastěji komunikuje a má k nim nejblíže. Ten bude s aplikací i pracovat. Kromě samotné aplikace mu také byly ukázány jednotlivé úlohy v programu RapidMiner a to, jak probíhá dolování v tomto prostředí. Ze třetí úlohy týkající se shlukování mu byla předána čísla zákazníků, kteří byli nalezeni v odlehlých hodnotách a na které by se měli zaměřit.

Firmě umožňuje vytvořená aplikace získat informace o svých produktech z pohledu dolování. Nalezené asociace mezi produky poslouží například k tomu, které produkty je vhodné nabízet společně. Uplatnění této znalosti se bude hodit například na konci roku, kdy zákazníkům zbývají peníze a firma tak může dle získaných asociací oslovit své zákazníky s nabídkou produktů. Predikční model Naive Bayes umožňuje určit, zda by si zákazník s konkrétní kombinací produktů daný produkt koupil.

Možné rozšíření aplikace by mohlo zahrnovat použití dalších klasifikačních metod. Zákazníci by se také mohli klasifikovat do více různých tříd, než jen do dvou – koupí/nekoupí. Toto rozdělení do tříd by se dalo odvíjet od jednotlivých skupin produktů, které zákazníci vlastní.

Aplikace je funkční, ale vzhledem k tomu, že je určena výhradně pro potřeby firmy, vyžaduje pro korektní běh přítomnost jejich dat. Z důvodu ochrany vlastnictví firemních dat nemohou být data odevzdána se zdrojovými soubory.

Literatura

[1] Bouckaert, R. R.; Frank, E.; Hall, M.; aj.: *WEKA Manual for Version 3-7-8*. Jan 2013, [Online; navštíveno 14.4.2019].
URL http://statweb.stanford.edu/~lpekelis/13_datafest_cart/WekaManual-3-7-8.pdf

[2] Brownlee, J.: *Improve Machine Learning Results with Boosting, Bagging and Blending Ensemble Methods in Weka*. Feb 2014, [Online; navštíveno 21.4.2019].
URL https://machinelearningmastery.com/improve-machine-learning-results-with-boosting-bagging-and-blending-ensemble-methods-in-weka/

[3] Demnag: *10 Popular Java Machine Learning Tools & Libraries*. Sep 2014, [Online; navštíveno 2.4.2019].
URL https://www.datasciencecentral.com/profiles/blogs/10-popular-java-machine-learning-tools-libraries

[4] doc.stable, W.: *Class Discretize*. [Online; navštíveno 1.4.2019].
URL http://weka.sourceforge.net/doc.stable/weka/filters/unsupervised/attribute/Discretize.html

[5] Han, J.; Kamber, M.: *Data Mining - Concepts and Techniques, 2nd Edition*. 2006, ISBN 978-1-55860-901-3.

[6] Hypertextbook: Chapter 2 Association Analysis,*Section 6 FP-Growth Algorithm*. [Online; navštíveno 5.5.2019].
URL http://www.hypertextbookshop.com/dataminingbook/public_version/contents/chapters/chapter002/section006/blue/page001.html

[7] Java: *Class RowFilter*. [Online; navštíveno 29.3.2019].
URL https://docs.oracle.com/javase/7/docs/api/javax/swing/RowFilter.html

[8] Java: *Class TableRowSorter*. [Online; navštíveno 29.3.2019].
URL https://docs.oracle.com/javase/7/docs/api/javax/swing/table/TableRowSorter.html

[9] Java: *Java (programovací jazyk)*. [Online; navštíveno 19.4.2019].
URL https://cs.wikipedia.org/wiki/Java_(programovac%C3%AD_jazyk)

[10] Mishra, A. K.; Pani, D. S. K.; Ratha, D. B. K.: *Association rule mining with Apriori and FP Growth using Weka. International journal of advanced technology in engineering and science*, ročník 3, č. 01, Sep 2015, ISSN 2348-7550, [Online; navštíveno 05.05.2019].

URL https://pdfs.semanticscholar.org/71cb/c6ed7bf4e04d5ad27981657f890b312cc048.pdf

[11] NetBeans: *Apache NetBeans*. [Online; navštíveno 19.4.2019].
URL https://netbeans.apache.org/

[12] RapidMiner: [Online; navštíveno 27.12.2018].
URL https://en.wikipedia.org/wiki/RapidMiner

[13] RapidMiner: Documentation - *Create Association Rules*. [Online; navštíveno 10.5.2019].
URL https://docs.rapidminer.com/latest/studio/operators/modeling/associations/create_association_rules.html

[14] Studio, R.: [Online; navštíveno 27.12.2018].
URL https://rapidminer.com/products/studio/

[15] Weka: *Weka 3: Data Mining Software in Java*. [Online; navštíveno 14.1.2019].
URL https://www.cs.waikato.ac.nz/ml/weka/index.html

[16] Weka-Wiki: *ARFF (stable version)*. [Online; navštíveno 21.4.2019].
URL https://waikato.github.io/weka-wiki/arff_stable/

[17] Witten, I. H.; Frank, E.; Hall, M. A.: *Data Mining: Practical machine learning tools and techniques*. 2005.

[18] Zendulka, J.; Bartík, V.; Lukáš, R.; aj.: *Získávání znalostí z databází*. Říjen 2009, [Online; navštíveno 20.12.2018].
URL https://wis.fit.vutbr.cz/FIT/st/cfs.php?file=%2Fcourse%2FZZN-IT%2Ftexts%2FZZN.pdf

Příloha A

Frekventované množiny

Velikost	Podpora	Produkt 1	Produkt 2	Produkt 3	Produkt 4	Produkt 5	Produkt 6
1	0.981	PAM					
1	0.908	ELD					
1	0.869	RNP					
1	0.658	HB*					
1	0.367	KZP					
1	0.252	PER					
1	0.152	UCT					
1	0.121	FAK					
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
2	0.908	PAM	ELD				
2	0.869	PAM	RNP				
2	0.658	PAM	HB*				
2	0.365	RNP	KZP				
2	0.120	UCT	FAK				
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
3	0.868	PAM	ELD	RNP			
3	0.324	RNP	HB*	KZP			
3	0.222	RNP	HB*	PER			
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
4	0.615	PAM	ELD	RNP	HB*		
4	0.324	PAM	RNP	HB*	KZP		
4	0.134	RNP	HB*	KZP	PER		
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
5	0.324	PAM	ELD	RNP	HB*	KZP	
5	0.134	ELD	RNP	HB*	KZP	PER	
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
6	0.134	PAM	ELD	RNP	HB*	KZP	PER

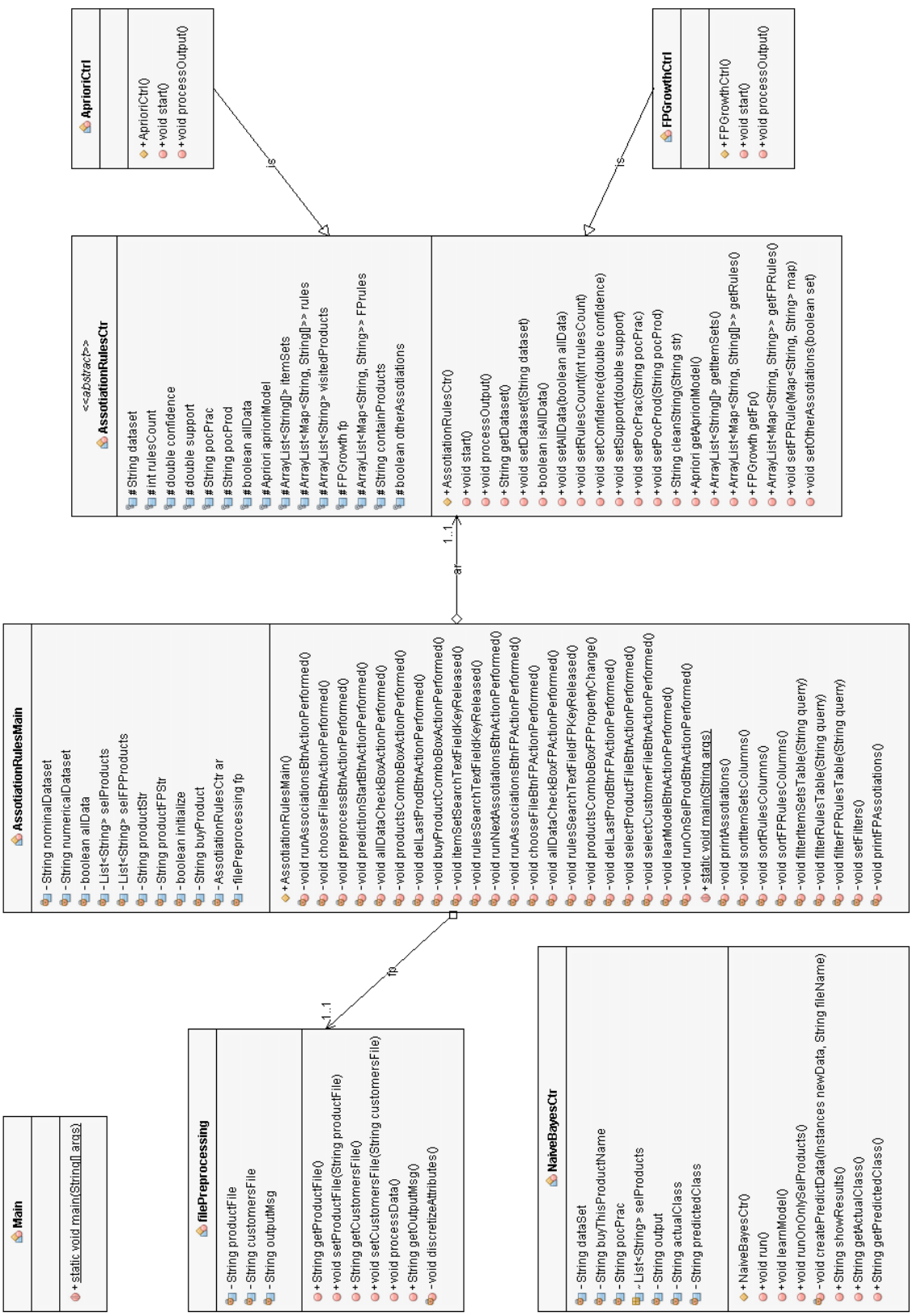
Tabulka A.1: Výčet některých frekventovaných množin nejčetnějších produktů

Velikost	Podpora	Produkt 1	Produkt 2	Produkt 3	Produkt 4	Produkt 5	Produkt 6
1	0.031	VZD					
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
2	0.031	PAM	VZD				
2	0.031	PER	VZD				
2	0.025	KZP	VZD				
2	0.020	HN0018	VZD				
2	0.020	STM	VZD				
2	0.015	HX0048	VZD				
2	0.014	HF0009	VZD				
2	0.012	HB*	KZP				
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
3	0.031	PAM	ELD	VZD			
3	0.031	RNP	PER	VZD			
3	0.011	HX0048	HN0018	VZD			
3	0.010	HX0048	STM	VZD			
3	0.010	HF0009	STM	VZD			
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
4	0.029	PAM	ELD	EB*	VZD		
4	0.031	PAM	RNP	PER	VZD		
4	0.007	STM	VZD	ePAM	ePER		
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
5	0.029	PAM	ELD	EB*	PER	VZD	
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
6	0.029	PAM	ELD	RNP	EB*	PER	VZD
6	0.024	PAM	RNP	EB*	KZP	PER	VZD

Tabulka A.2: Výčet některých frekventovaných množin zahrnující produkt VZD

Příloha B

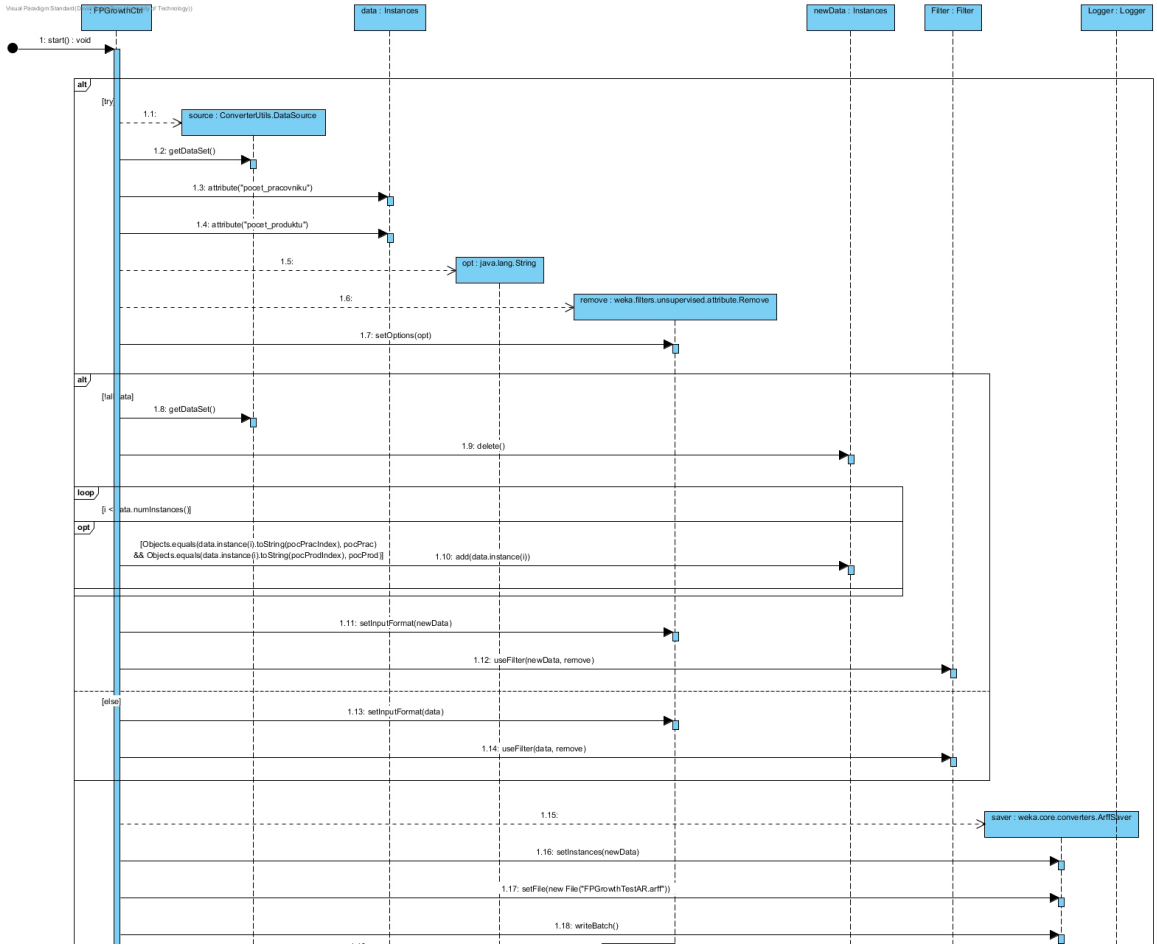
Diagram tříd



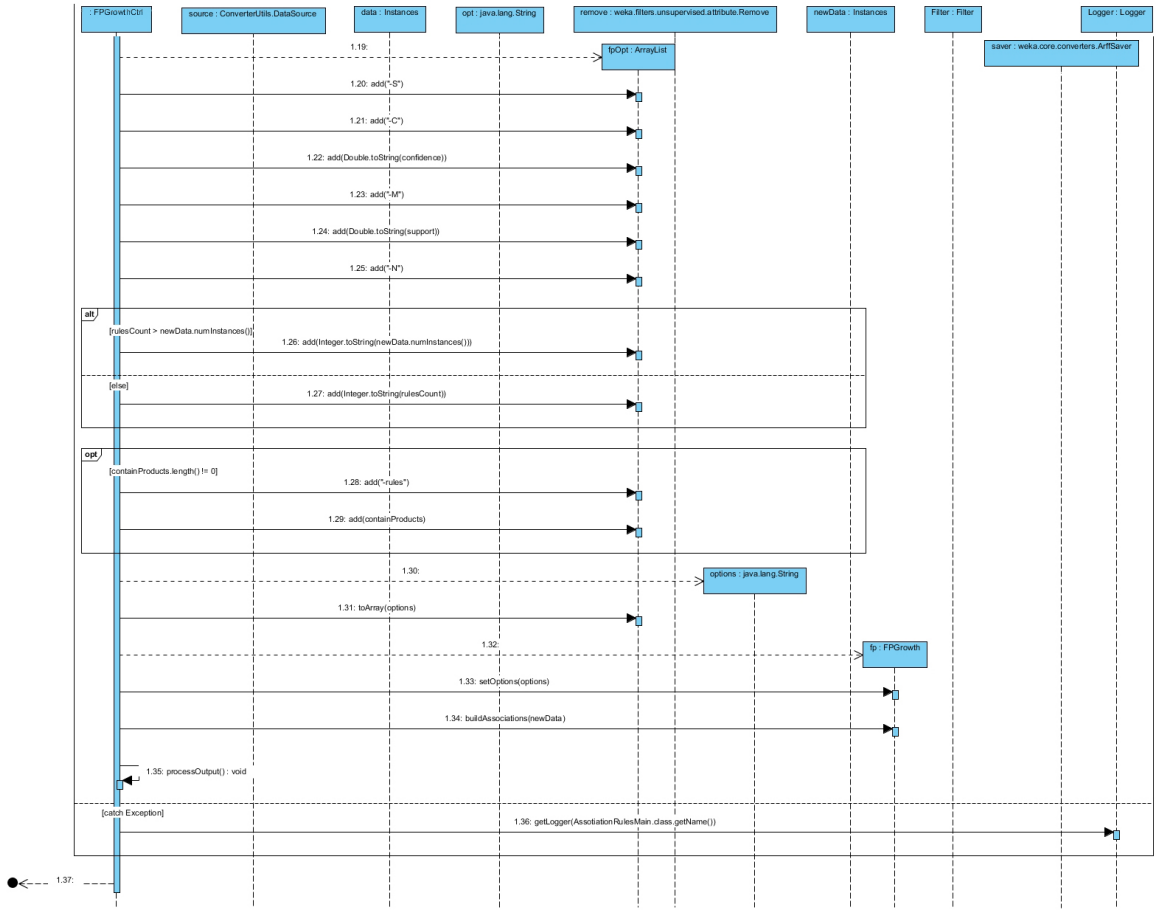
Obrázek B.1: Diagram tříd

Příloha C

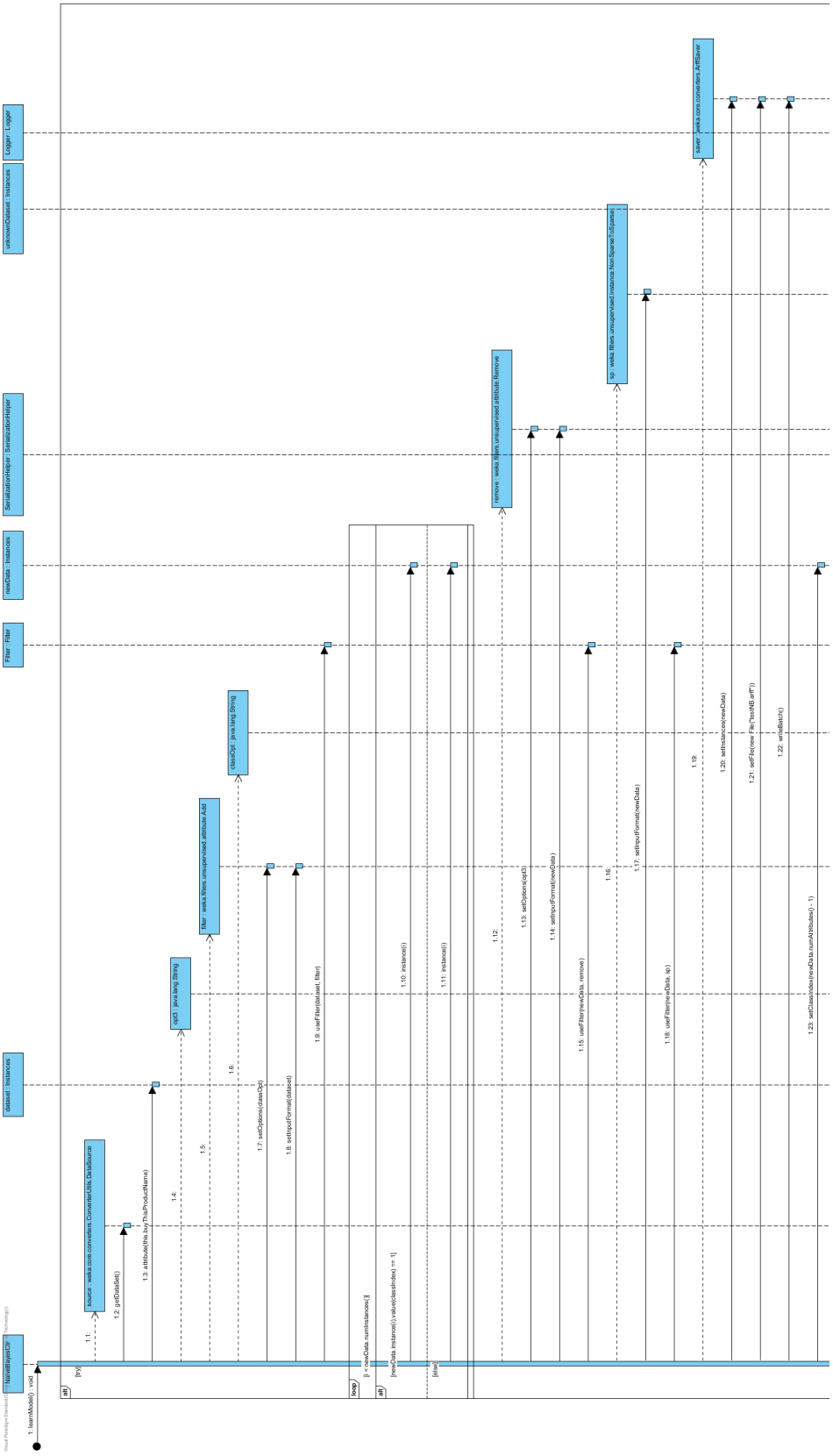
Sekvenční diagramy



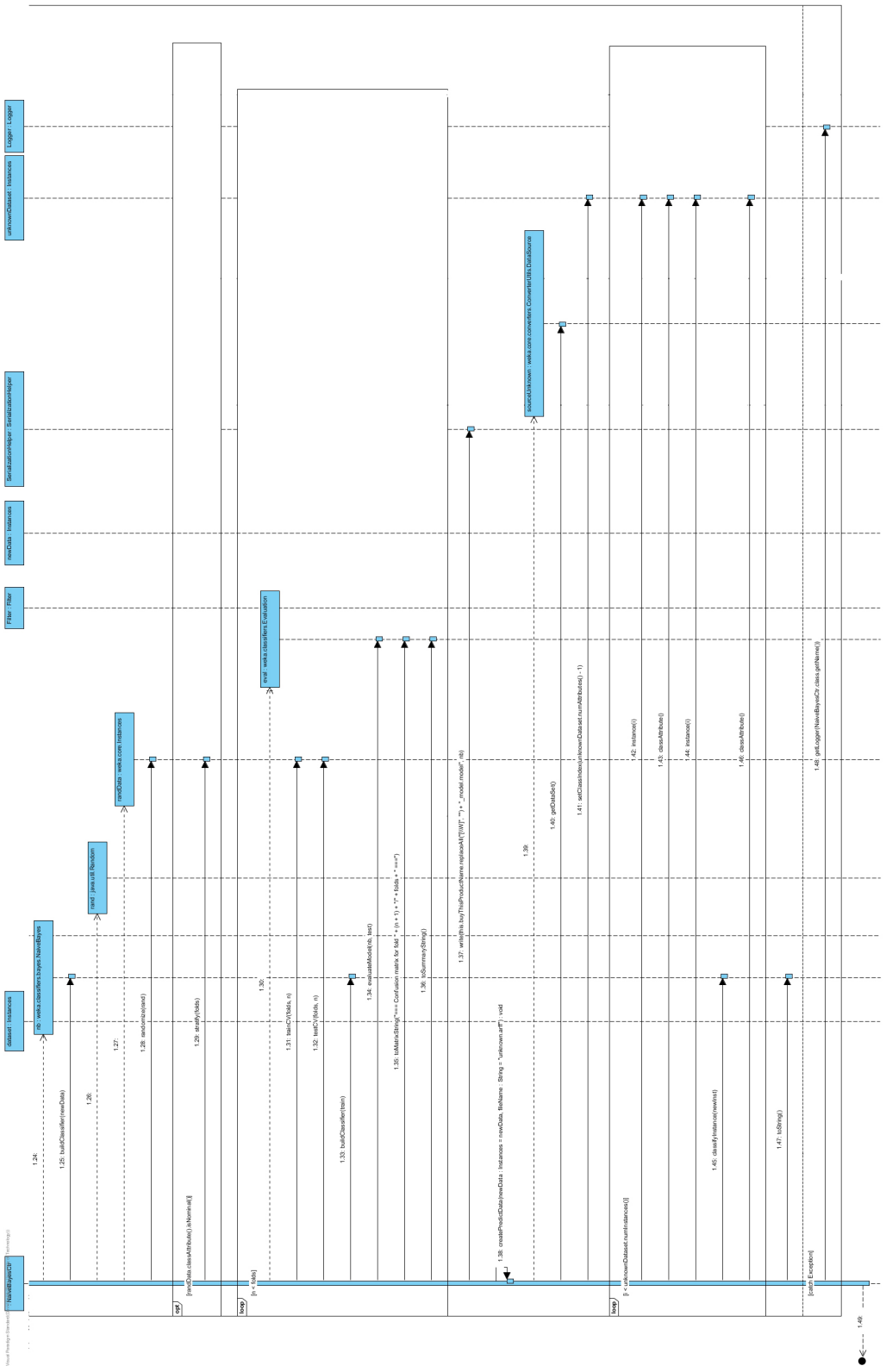
Obrázek C.1: Sekvenční diagram metody start() - FP Growth



Obrázek C.2: Sekvenční diagram metody start() - FP Growth (pokračování)



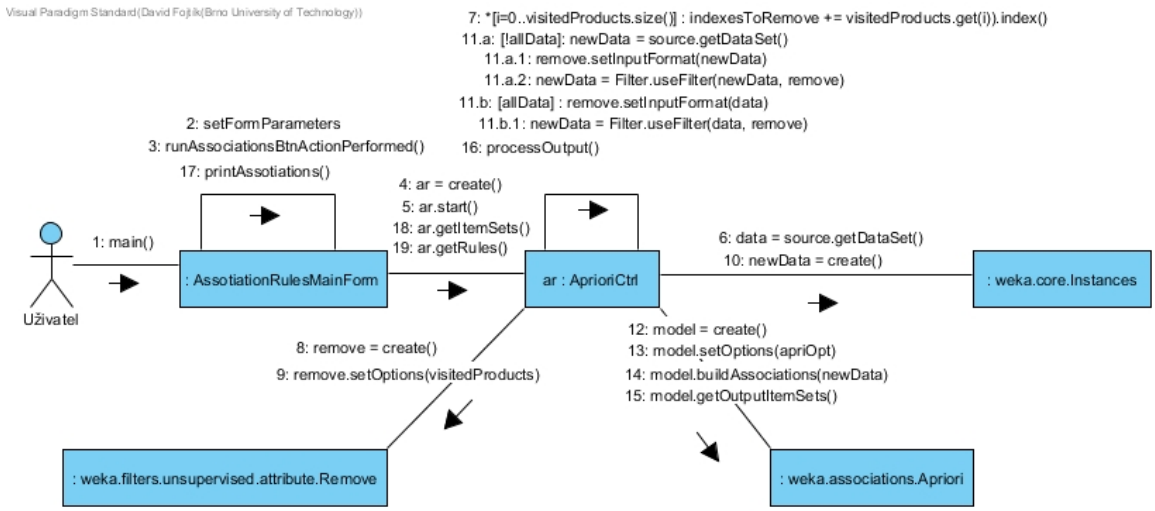
Obrázek C.3: Sekvenční diagram metody learModel() - Naive Bayes



Obrázek C.4: Sekvenční diagram metody learModel() - Naive Bayes (pokračování)

Příloha D

Diagram komunikace



Obrázek D.1: Diagram komunikace hledání dalších asociačních pravidel - Apriori

Příloha E

Obsah přiloženého CD

Na CD dodaném k této diplomové práci se nacházejí následující adresáře:

```
+lib // Knihovna Weky
  |__ weka.jar
+nbproject // Projekt pro NetBeans
+src // Zdrojove soubory
  |__ Control
      |__ AprioriCtrl.java
      |__ AssotiationRulesCtr.java
      |__ filePreprocessing.java
      |__ FPGrowthCtrl.java
      |__ NaiveBayesCtr.java
  |
  |__ dp
      |__ Main.java
  |
  |__ gui
      |__ AssotiationRulesMain.form
      |__ AssotiationRulesMain.java
+dp.pdf // Diplomova prace ve formatu pdf
+LaTeX // Zdrojove kody diplomove prace v LaTeXu
```