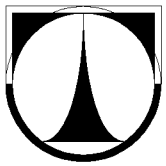
TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií



DIPLOMOVÁ PRÁCE

Liberec 2011 **Bc. Pavel Karol**

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií

Studijní program: N2612 – Elektrotechnika a informatika
Studijní obor: 1802T007– Informační technologie

Datamining ve výuce IT

Datamining in IT educations

Diplomová práce

Autor: **Bc. Pavel Karol**
Vedoucí práce: RNDr. Klára Císařová, Ph.D.
Konzultant: Ing. Martina Černíková, Ph.D.

V Liberci 3. 1. 2011

Originál zadání práce

Prohlášení

Byl(a) jsem seznámen(a) s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 o právu autorském, zejména § 60 (školní dílo).

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o užití mé diplomové práce a prohlašuji, že **s o u h l a s í m** s případným užitím mé diplomové práce (prodej, zapůjčení apod.).

Jsem si vědom(a) toho, že užít své diplomové práce či poskytnout licenci k jejímu využití mohu jen se souhlasem TUL, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

Diplomovou práci jsem vypracoval(a) samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce a konzultantem.

Datum

Podpis

Poděkování

Na tomto místě bych chtěl poděkovat svojí vedoucí RNDr. Kláře Císařové, Ph.D. za poskytnutí odborné pomoci, za trpělivost při vedení mé práce a také za zapůjčení počítače s licencí softwaru, potřebného k vytvoření praktické části této práce. Děkuji také své rodině a přátelům, kteří mě ve studiu podporovali a pomáhali mi.

Abstrakt

Tato diplomová práce zahrnuje koncept dataminingu a použití analytických modelů v program SPSS PASW Modeler (dříve Clementine). Obsahuje množství teoretických podkladů k jednotlivým vybraným modelům, jejichž funkce jsou založeny především na termínech z oblasti teorie informace a statistiky. Je rozdělena do pěti přehledných kapitol. Přílohy obsahují příklady k jednotlivým modelům vytvořené v systému Clementine, popis jednotlivých datových proudů a funkce použitých uzlů.

Klíčová slova: Datamining, Rozhodovací stromy, Asociační pravidla, Neuronové sítě, PASW Modeler

Abstract

This diploma thesis comprises of the data mining concept and the use of analytical models in the SPSS PASW Modeler software (formerly called Clementine). It also includes an amount of theoretical bases for each of the selected models, functions of which are based mostly on the technical terms from within the statistical and information theories. It is divided into five well-arranged chapters. The appendices include examples of the models created using the Clementine software system, a description of each data stream and the role of the nodes used.

Keywords: Data mining, Decision trees, Association Rules, Neural Networks, PASW Modeler

Obsah

Úvod

Datamining je rozsáhlé velmi aktuální téma, které v sobě obsahuje několik vědních oborů. Střetává se zde matematika, teorie informace, statistika a umělá inteligence, vše aplikováno především v oblasti databázových systémů, to ale nevylučuje i jiné rozsáhlé datové zdroje. Metody DM se používají za účelem zjišťování dosud neznámých, skrytých informací v datech, které slouží zejména k podpoře marketingu. Na ústavu MTI FM se připravuje nový předmět, který by měl studentům přiblížit tuto problematiku a rozšířit tak nabídku informatických předmětů. V současné době je k dispozici jen málo materiálů k tomuto tématu v češtině, proto část této diplomové práce bude sloužit jako studijní materiál. Hlavním cílem této diplomové práce bylo popsat vybrané analytické metody dataminingu a připravit k těmto metodám několik příkladů. Pro vybrané metody byly sestavené modely v prostředí SPSS PASW Modeler (dříve Clementine®), které má MTI k dispozici. Byly tak připravené příklady datových proudů, jejichž výsledkem jsou popsané analytické modely. Dalším cílem bylo vytvořit vhodný manuál pro výuku SPSS PASW Modeleru, protože firemní materiál se ukázal jako nedostačující.

Metodika

Práce byla vytvořena na základě studia několika zdrojů: z knih Datamining, praktický průvodce dolováním dat pro efektivní prodej, cílený marketing a podporu zákazníků (CRM) od Olivie Parr Rud; Dobývání znalostí z databází od Petra Berky; Statistika pro ekonomy od Richarda Hindlse, Stanislavy Hronové a Jana Segera; Databáze: datové sklady, OLAP a dolování dat s příklady v Microsoft SQL Serveru a Oracle od Luboslava Lacka. Při popisu jednotlivých modelů byla využita firemní dokumentace k programu PASW Modeler 13 od dodavatele tohoto softwaru, firmy Predictive Analytics Software. Dalším významným zdrojem informací byl internet a to především za účelem ověřování a upřesňování informací, např. www.spss.cz, http://www.msps.cz/data-mining/.

1. Datamining

1.1 Historie

Lidstvo se zabývá získáváním a uchováváním informací po celá tisíciletí. Důkazem toho může být vynález písma již na počátku starověku.

S přibývajícím množstvím dat bylo zapotřebí přikročit k nějakému automatickému mechanismu, který by tato data byl schopný uchovat a zpracovat rychle. Mezi starší metody zpracování dat patří například identifikace datových vzorů za použití Bayesova teorému, datující se do 18. století, anebo často používaná regresivní analýza (19. století). S rozvojem výpočetní techniky v 50. letech minulého století se začala rozvíjet i tato disciplína. Nárůst množství a komplexnosti datových zdrojů vedl ke zvětšení objemu nepřímých dat, která byla získávána v průběhu tohoto procesu. Tento problém byl základním stavebním kamenem pro nové objevy v oblasti počítačové vědy, jako např. neuronových sítí, shlukové analýzy, genetických algoritmů (1950), rozhodovacích stromů (1960), nebo termínu strojového učení.

O samotném termínu datamining, nebo též ekvivalentní varianty dobývání znalostí z databází (Knowledge Discovery in Databases, dále jen KDD), se začalo ve vědeckých kruzích mluvit počátkem 90. let minulého století. Poprvé zazněl tento termín na mezinárodní konferenci umělé inteligence IJCAI v roce 1989 a dále na konferenci americké asociace umělé inteligence v roce 1991. Zpočátku se pro tuto oblast razily nejrůznější názvy:information harvesting, data archeology, data distillation. Nakonec ale převládla hornická metafora dobývání znalostí z databází a dolování z dat (datamining). Po jisté době tápání se ustálilo i chápání KDD jako interaktivního a iterativního procesu tvořeného kroky selekce, předzpracování, transformace, vlastního „dolování“ a interpretace. [4]

1.2 Definice dataminingu

(citace z [1]) Dobývání znalostí z databází je pojmem z oblasti Business Intelligence a definuje se jako „netriviální získávání implicitních, dříve neznámých a potenciálně užitečných informací z dat“. Primárními zdroji pro realizaci pokročilého zpracování dat byly:

1) Databázové technologie, které představují osvědčený prostředek, jak uchovávat rozsáhlá data a vyhledávat v nich informace.

2) Statistika, která vždy představovala osvědčený prostředek, jak modelovat a analyzovat asociace ukryté v datech.

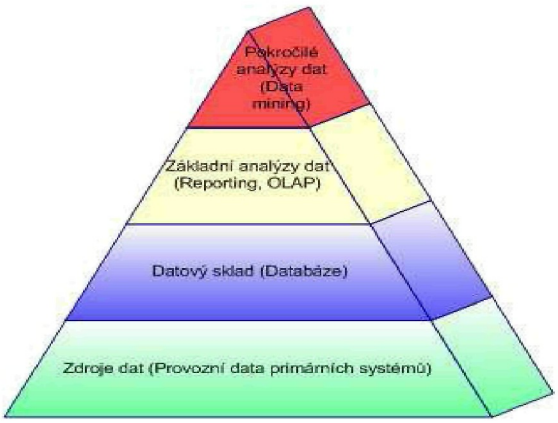
Dlouhou dobu se tyto disciplíny vyvíjely nezávisle. Objevil se zájem finančně silných subjektů o aplikace, které by byly schopné propojit tyto dvě disciplíny, a následně byl stimulem pro vznik aplikací schopných „dobývání znalostí“ z databází.

Datamining je pojem zastřešující širokou škálu technik používaných dnes v řadě odvětví. Vzhledem k zostřené konkurenci v boji o zisky a tržní podíly v marketingové aréně se dolování dat stalo nezbytnou praxí vedoucí k udržení konkurenční schopnosti firem. Obvykle každý komerční i nekomerční subjekt shromažďuje data ve svých vlastních systémech. Tato data slouží k následné analýze, která může pomoci objevit trendy, na základě nichž lze činit důležitá marketingová a obchodní rozhodnutí. [4]

Datamining je momentálně jednoznačně nejrychleji rostoucí segment Business Intelligence. Je to průnik umělé inteligence do oblasti databází. Databázové tabulky obsahují miliony až miliardy záznamů, které jsou různě uspořádané a členěné. Datamining umožňuje v těchto údajích vyhledávat vzory informací. [1]

1.3 Business Intelligence

Pod označením Business Intelligence si lze představit především výkonné analytické a vykazovací nástroje, které umožňují využít firemní data nejen k analýze již existujících jevů, ale také k predikcím budoucího vývoje. Většina firemních systémů je zahlcená daty, ze kterých se relevantní informace získávají jen velmi obtížně. BI nástroje uživatelům umožní pracovat s daty z různých úhlů pohledu a jsou cestou ke konkurenční výhodě. Sběrem a vyhodnocováním dat jsou získávány relevantní informace, které podporují plánování, rozhodování a řízení společnosti. Tento proces zobrazuje BI pyramida na obr. 1.1. Základnou této pyramidy jsou provozní data primárních systémů. Na základě těchto dat se vytváří struktura databáze. Uložením dat do této struktury vzniká datový sklad, který je základním stavebním kamenem pro budoucí analýzy. Ať už jsou to základní analýzy OLAP nebo pokročilé analýzy dataminingu, využívající např. rozhodovacích stromů nebo neuronových sítí. [3]



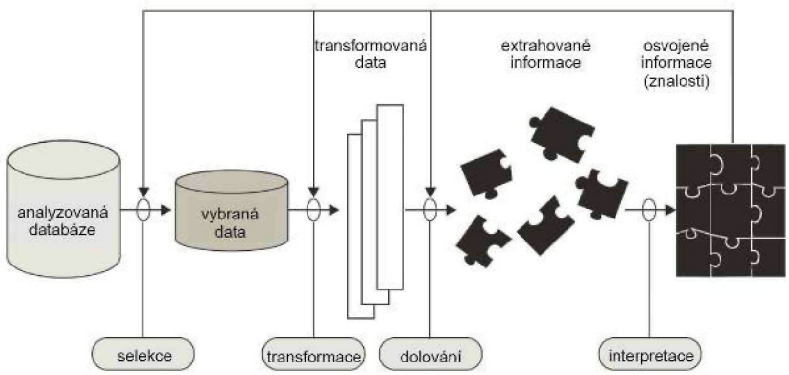
1.1 Pyramida Business Intelligence (převzato z [5])

1.4 Proces dobývání znalostí z databází

Impulsem pro zahájení procesu KDD je nějaký reálný problém. Cílem je získat co nejvíce relevantních informací vhodných k řešení daného problému. Existují dva úhly pohledu na tuto problematiku. Technologický pohled a manažerský pohled. [1]

1.4.1 Technologický pohled

Z technologického pohledu se v procesu KDD klade důraz na přípravu dat a interpretaci výsledných znalostí. Data jsou obvykle uložena ve složité struktuře databáze resp. datového skladu. Při přípravě dat se z této struktury vytváří jedna tabulka obsahující relevantní údaje o sledovaných objektech. Např. hodnoty atributů u klientů banky. Interpretace nalezených znalostí se poté hodnotí z pohledu koncového uživatele. Schéma technologického pohledu na KDD je zobrazeno na obr 1.2



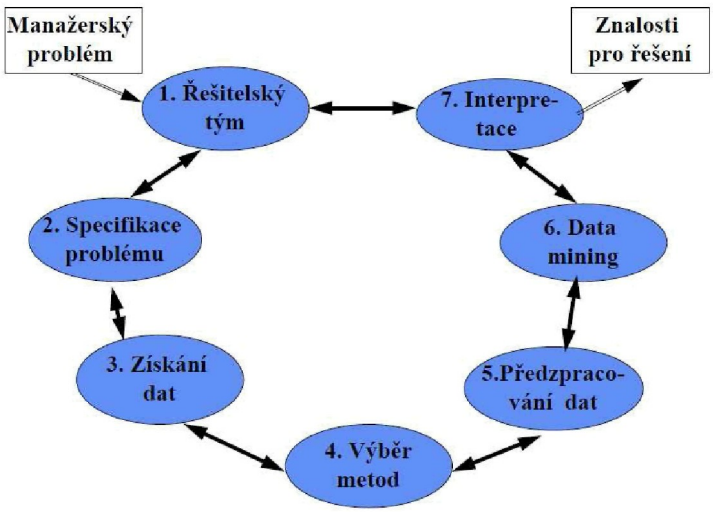
1.2 Technologický pohled na proces KDD (převzato z [6])

1.4.2 Manažerský pohled

Manažeři technologické problémy neřeší, musí zajistit chod procesu KDD, aby mohlo dojít k vyřešení zadaného problému. Proces KDD z manažerského pohledu by se dal popsat v sedmi krocích:

1. Sestavení řešitelského týmu
2. Specifikace problému
3. Získání všech dostupných dat
4. Výběr vhodné analýzy dat
5. Předzpracování dat
6. Datamining
7. Interpretace výsledků

Pokud nejsme spokojeni s průběhem procesu, můžeme jednotlivé kroky opakovat, přeskakovat apod. Nejlépe je tento proces vidět na schématu viz obrázek 1.3 [7]



1.3 Manažerský pohled na proces KDD (převzato z [7])

1.5 Úlohy dataminingu

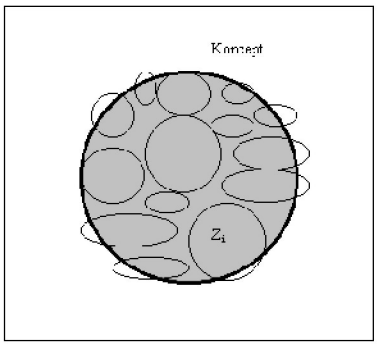
Úlohy dataminingu jsou interpretovány a reprodukovány v různých formách a dají se rozdělit do tří základních skupin.

1. Klasifikace nebo predikce

2. Deskripce
3. Hledání „nuggetů“[1]

1.5.1 Klasifikace, predikce

Klasifikace popř. predikce má za cíl nalézt použitelné znalosti pro hodnocení nových případů. Požadavkem této úlohy je zisk většího množství znalostí, které co nejlépe odpovídají danému konceptu. Klade se větší důraz na pokrytí dat, než na jednoduchost, přesněji řečeno připouští se větší množství srozumitelných dílčích znalostí viz obr. 1.4



1.4 Klasifikace nebo predikce [7]

Proces klasifikace se skládá ze dvou kroků:

1. Učení: tvorba klasifikačního modelu schopného klasifikovat data pomocí tréninkových dat (vzorků dat, u nichž známe výsledek klasifikace, tj. třídu, do které patří),
2. Vlastní klasifikace: použití modelu pro klasifikaci nových dat (jejich zařazení do tříd)

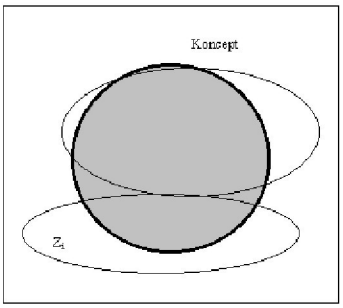
Jako predikce je označován proces určení dodatečných, případně chybějících, hodnot atributů analyzovaného záznamu. Jedním z druhů predikce je výše zmiňovaná vlastní klasifikace, kdy se určuje dodatečný atribut, představující třídu objektu (atribut diskrétního charakteru).

Významnějším druhem predikce je regrese, kdy jsou pomocí modelu vzniklého klasifikací dopočítány číselné hodnoty některého atributu spojitého charakteru. [7]

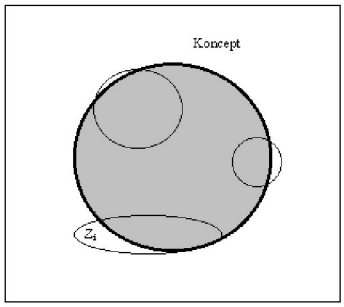
[1] zajímavé vzory, které se mohou týkat třeba jen malých podmnožin vstupních dat

1.5.2 Deskripce a hledání nuggetů

Cílem deskripce, neboli popisu, je nalézt dominantní strukturu, nebo vazby, které jsou skryté v daných datech. Požadujeme srozumitelné znalosti pokrývající daný koncept. Dáváme přednost menšímu množství méně přesných znalostí. Pokud hledáme „nuggety", zajímají nás nové a překvapivé znalosti, které nemusí pokrývat daný koncept. Tyto typy úloh jsou znázorněny na obr. 1.5 a 1.6



Obr. 1.5 Deskripce (převzato z [7])



Obr. 1.6 Nuggety (převzato z [7])

Úlohy dataminingu lze nalézt v celé řadě aplikační oblastí, např.

- Segmentace a klasifikace klientů banky (rozpoznání problémových nebo naopak bonitních klientů)
- Predikce vývoje kurzů akcií
- Predikce spotřeby elektrické energie
- Analýza příčin poruch v telekomunikační síti
- Analýza důvodů změny poskytovatele služeb (internet, mobilní operátor)
- Segmentace a klasifikace klientů pojišťovny
- Určení příčin poruch automobilů
- Rozbor databáze pacientů nemocnice
- Analýza nákupního košíku (Market Basket Analysis) [1]

1.6 Metodiky

Metodiky mají za cíl poskytnout uživatelům jednotný rámec pro řešení úloh z oblasti dataminingu. Za vznikem některých metodik stojí producenti programových systémů, jiné vznikají ve spolupráci výzkumných a komerčních institucí jako „softwarově nezávislé". Mezi

metodiky vázané na programové systémy patří metodika 5A firmy SPSS nebo metodika SEMMA firmy SAS. Do druhé skupiny patří metodika CRISP-DM. [8]

1.6.1 Metodika 5A

Název metodika 5A od firmy SPSS je akronymem pro jednotlivě prováděné kroky procesu KDD. Tyto kroky jsou:

- Assess – posouzení potřeb projektu
- Access – shromáždění potřebných dat
- Analyze – provedení analýz
- Akt – přeměna znalostí na akční znalosti
- Automate – převedení výsledků analýzy do praxe [8]

1.6.2 Metodika SEMMA

Tato metodika vychází ze softwarového produktu firmy SAS, Enterprise miner. Zkratka SEMMA je opět akronymum jednotlivých kroků.

- Sample (vybrání vhodných objektů)
- Explore (vizuální explorace a redukce dat)
- Modify (seskupování objektů a hodnot atributů; datové transformace)
- Model (analýza dat: neuronové sítě, rozhodovací stromy, statistické techniky, asociace a shlukování)
- Assess (porovnání modelů a interpretace) [8]

1.6.3 Metodika CRISP-DM

Metodika CRISP-DM (CRoss-Industry Standard Process for Datamining) se snaží nalézt univerzálně použitelný postup pro dolování dat. Z použité literatury lze také usuzovat, že terminologie uváděná v rámci této metodiky tvoří v oboru již standard, alespoň ve své anglické jazykové mutaci. Metodika CRISP-DM uvádí následující etapy:

- porozumění problematice (Business Understanding)
- porozumění datům (Data Understanding)

- příprava dat (Data Preparation)
- modelování (Modeling)
- vyhodnocení výsledků (Evaluation)
- využití výsledků (Deployment) [8]

Porozumění problematice je fáze, která je zaměřena na pochopení cílů úlohy a požadavků na řešení formulovaných z manažerského hlediska. Manažerská formulace musí být následně převedena do zadání úlohy pro dobývání znalostí z databází. V této fázi se rovněž provádí inventura zdrojů (datových výpočetních i lidských). Hodnotí se možná rizika, náklady a přínos použití metod KDD a stanovuje se předběžný plán prací.

Fáze porozumění datům začíná prvotním sběrem dat. Následují činnosti, které umožní získat základní představu o datech, která jsou k dispozici. Obvykle se zjišťují různé deskriptivní charakteristiky dat (četnosti hodnot různých atributů, průměrné hodnoty, minima, maxima apod.) s výhodou se využívají i různé vizualizační techniky.

Příprava dat zahrnuje činnosti, které vedou k vytvoření datového souboru, který bude zpracováván jednotlivými analytickými metodami. Tato data by měla obsahovat údaje význačné pro danou úlohu a měla by mít podobu, která je vyžadována vlastními analytickými algoritmy. V této fázi dochází k selekci, čištění, transformaci, vytváření, integrování a formátování dat. Je to nejpracnější část celé řešené úlohy. Jednotlivé úkony jsou obvykle prováděny opakovaně.

Pokud jsme spokojeni s daty, která jsme připravili v předchozí fázi, můžeme přikročit k modelování. V této fázi jsou nasazeny analytické metody (algoritmy pro KDD). Je třeba vybrat nejvhodnější metodu, neboť k řešení dané úlohy existuje celá řada metod, které by se daly použít. Doporučuje se použít více metod a jejich výsledky kombinovat. Použití analytických metod může vést k potřebě modifikovat data, a tedy k návratu k datovým transformacím předchozí fáze. Součástí této fáze je též ověřování nalezených znalostí z pohledu metod dobývání znalostí např. testování klasifikačních znalostí na nezávislých datech.

Pokud jsme se dostali do fáze vyhodnocení výsledků, tak jsme se dopracovali do stavu nalezení znalostí, které se zdají být v pořádku z hlediska metod dobývání znalostí. Dosažené výsledky je ale ještě třeba vyhodnotit z pohledu manažerů, zda byly splněny cíle formulované

při zadání úlohy. Na závěr této fáze by mělo být přijato rozhodnutí o způsobu využití výsledků.

Vytvořením vhodného modelu práce na zadání nekončí. Dokonce i v případě, že řešenou úlohou byl pouze popis dat, je potřeba získané znalosti převést do podoby použitelné pro zákazníka. Výsledky mohou být sepsány do závěrečné zprávy, nebo může být zaveden systém (hardwarový, softwarový, organizační) pro automatickou klasifikaci nových případů. Samozřejmě záleží na typu úlohy. Je důležité, aby zákazník pochopil výsledky analýzy tak, aby mohly být využívány efektivně.

Jednotlivé kroky procesu jsou různě časově náročné a mají také různou důležitost pro úspěšné vyřešení dané úlohy. Nejdůležitější fáze je porozumění problému, která má 80% významu a zabere 20% času. Časově nejnáročnější je fáze přípravy dat, u které je poměr přesně obrácený. Vlastní analýzy překvapivě znamenají nejkratší úsek procesu, v procentech 5% času, 5% významu. [1]

1.7 Typy zdrojů dat

Stejně jako datamining podporuje širokou škálu úloh, tak i rozmanitost datových zdrojů je poměrně velká. Data získaná pro proces KDD by se dala rozdělit do tří skupin, tedy získaná z:

- databázových systémů - relační databáze, OLAP(on-line Analytical Processing) a datové sklady (Data Warehouse).
- metod strojového učení
- statistických analýz – kontingenční tabulky, regresní analýza, diskriminační analýza, shluková analýza (samostatná skupina, která patří již mezi analytické metody)

1.7.1 Relační databáze

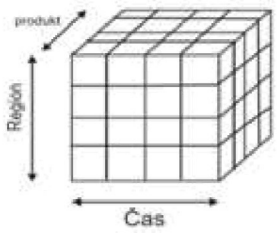
Ve většině případů se jako zdroj pro proces KDD používají relační databáze. U těchto databází se předpokládá vzájemná nezávislost záznamů z hlediska pořadí v databázi. Pojem relační databáze byl definován již v roce 1969. Dr. Edgar F. Codd vymyslel model databáze založený na relační algebře. Byl to první databázový systém, kde teorie a matematické

algoritmy předcházely programátorskou praxi pro databázovou technologii. Návrh datových struktur relačních databází byl a dodnes je spojen s ER diagramy, kterými se popisují data a vztahy v těchto datech, aby bylo možné modelovat relevantní datovou realitu.

Pro práci s daty, uloženými v databázových systémech, vytvořila v 70. letech minulého století firma IBM dva jazyky QBE (Query by example) a SQL (Structured query language). V následujících letech IBM přišla o primát v oblasti technologií relačních databází z důvodů manažerských pochybení. Do dnešních dnů je databázová realita převážně relační a jen pomalu se mění v tzv. postrelační, či objektové databázové technologie. Pro problémy související z dataminingem je toto celkem nevýznamné. Diskuse o příčinách a prognóze vývoje databázového světa by překročila rámec rozsahu a zadání této diplomové práce.

1.7.2 OLAP

Základem OLAP (On-Line Analytical Processing) je mnohorozměrná tabulka, nazývaná *datová krychle* (data cube). Příklad takové krychle je zobrazen na obr. 1.7. Jednotlivě sledované atributy tvoří dimenze krychle, buňky krychle pak odpovídají jednotlivým záznamům v databázi. Tento způsob uložení umožňuje různé pohledy na data. Nevýhodou je plýtvání místem, jelikož řada buněk v krychli zůstane prázdná. Práce s krychlí spočívá v různém natáčení (pivot), provádění řezů (slice), výběru určitých částí (dice) a zobrazování různých agregovaných hodnot. Velmi často lze hodnoty atributů sdružovat do hierarchií, např. město -> okres -> region -> země. Tyto hierarchie zobrazují různé úrovně podrobností (granularity) dat. [2]



Obr. 1.7 Ukázka datové krychle

OLAP nabízí uživatelům jak flexibilitu při práci s daty, tak i intuitivní ovládání. Typické pro OLAP jsou možnosti vizualizace. Grafické rozhraní umožňuje uživateli nahlížet na data v numerické formě, ale i ve formě nejrůznějších grafů. Pro OLAP systémy definoval koncept Dr. E. F. Codd. Tento koncept má následující pravidla:

- *Multidimenzionální konceptuální pohled* – OLAP by měl vytvořit uživateli multidimenzionální model tak, aby tento model mohl být využit na konkrétní analýzy vzhledem k potřebám uživatele
- *Transparentnost* – Technologie OLAP, podřízená databáze a architektura výpočtů by mělo být pro uživatele přehledné. K přehlednosti slouží různé klientské aplikace např. MS Excel (napojený na server OLAP).
- *Dostupnost* – V systému OLAP je potřeba zajistit přístup jen k údajům, které jsou potřeba pro analýzu, bez ohledu na to, odkud tyto údaje pocházejí.
- *Konzistentní vykazování* – Uživatel by neměl pocítit snížení výkonu systému při zvětšování objemu datového skladu
- *Flexibilní vykazování* – Schopnost uspořádaní řádků, sloupců a buněk do takové formy, která umožní analýzu a intuitivní vizuální prezentaci analytických sestav
- *Generická dimenzionalita* – Každá dimenze údajů musí být v operačních schopnostech i ve struktuře ekvivalentní
- *Podpora více uživatelů* – Systém OLAP musí podporovat skupinu uživatelů, kteří pracují ve stejný okamžik na konkrétních modelech
- *Intuitivní manipulace s údaji* – Uživatelské rozhraní by mělo obsahovat všechny funkce typu „drag and drop“, „click and show“ v buňkách krychle
- *Neomezené dimenze a úrovně agregace* – Systém OLAP by neměl za žádných okolností zavádět umělá omezení počtu dimenzí nebo úrovní agregace. Nároky uživatelů se odvíjejí v závislosti na požadavcích podnikání
- *Neomezené křížové dimenzionální operace* – OLAP musí umět rozeznat dimenzionální hierarchie a automaticky vykonat asociované kumulované kalkulace v rámci dimenzí i mezi nimi
- *Architektura klient-server* – Systém OLAP musí odpovídat architektuře klient-server, s přihlédnutím k maximálnímu výkonu a flexibilitě.
- *Dynamické ošetření řídkých matic* – Systém OLAP by měl být schopen adaptace svého fyzického schématu na konkrétní analytický model, který optimalizuje ošetření řídkých matic. Je nutné, aby udržel požadovanou úroveň výkonu. [2]

OLAP se liší od dataminingu tím, že získává z dat pouze sumární charakteristiky na zvolené podrobnosti pohledu, zatímco při dataminingu hledáme v datech nějaké zajímavé souvislosti. OLAP přináší odpovědi na konkrétní, přesně specifikované otázky, ale sám o sobě nic „neobjevuje". [2]

1.7.3 Datový sklad

Zatímco OLAP představuje nástroj pro analýzu (a vizualizaci) dat o firmě, datový sklad představuje místo, kde jsou analyzovaná data uložena. Koncept datového skladu byl zformulován v 80. letech minulého století W. H. Inmonem. *„Datový sklad je podnikově strukturovaný depozitář subjektově orientovaných, integrovaných, časově proměnlivých, historických dat použitých na získávání informací a podporu rozhodování. V datovém skladu jsou uložena atomická a sumární data."* (citace z [2])

- *Subjektová orientace* – Data v datovém skladu jsou orientována na subjekt. To může být např. dodavatel, zákazník. Vůbec nehraje roli, z jakého systému jsou tato data získána. Opak subjektové orientace je aplikační orientace.
- *Integrovanost* – Toto je velmi důležitá a obtížná úloha, která má vliv na celý datový sklad. Údaje o konkrétním předmětu se ukládají pouze jednou. Je zapotřebí nalézt jednotnou terminologii a konzistentní jednotky. Pokud data nejsou konzistentní, ztrácí datový sklad význam.
- *Časová proměnlivost* – Údaje se ukládají do datového skladu jako časové snímky. To znamená, že data jsou platná jen po určitou dobu. Pokud snímek v datovém skladu získá čas do klíčového atributu, již dále nemůže být modifikován. Doba platnosti se u datových skladů pohybuje v rámci let. Naproti tomu operační databáze má tuto platnost pouze několik dní.
- *Neměnnost* – S údaji již uloženými v datovém skladě se nedají dělat jiné operace, než je čtení a vyhledávání. U operačních databází se mohou tato data libovolně měnit. [2]

Datový sklad obsahuje obvykle operační data uložená v daném okamžiku, dále starší (historická) operační data, souhrny na různých úrovních abstrakce a tzv. metadata, které zachycují informace o datech. Jako příklad použijme datový sklad obsahující informace o prodeji nějakých výrobků: Současná detailní data mohou být údaje o prodeji v roce 2010. Historická operační data mohou být údaje za rok 2007-2009. Středně sumarizovaná data

mohou udávat souhrnný prodej po skupinách výrobků (nebo po okresech) za jednotlivé týdny. Silně sumarizovaná data mohou obsahovat souhrnný měsíční prodej za jednotlivé kraje. Metadata mohou být informace o tom, jaké okresy tvoří jednotlivé kraje, případně které výrobky patří do jednotlivých skupin apod. [2]

1.8 Analytické metody dataminingu

Samotným jádrem procesu dobývání znalostí z databází jsou analytické metody. Tyto metody se aplikují na předem zpracovaná data s cílem získat z těchto dat užitečné znalosti. Za předpokladu, že vycházíme z představy, že si navzájem podobné příklady (neboli příklady téže třídy) vytvářejí shluky v prostoru atributů, pak znalosti, které chceme získat, jsou právě tyto shluky. Získané znalosti se poté musí nějakým způsobem prezentovat. Způsob reprezentace je však poměrně rozmanitý. Například to mohou být reprezentativní příklady-etalony, nebo to mohou být funkce přiřazené jednotlivým shlukům. To, jak jsou jednotlivé metody reprezentovány, ovšem není jediný rozdíl. Každý, kdo vybírá vhodnou metodu pro analýzu dat, si musí položit následující otázky:

- Pro jaký typ úlohy je použitá metoda vhodná?
- Jak složité shluky dokáže reprezentovat?
- Jak srozumitelné jsou nalezené znalosti pro uživatele?
- Jak jsou tyto znalosti efektivní při klasifikaci nových případů?
- Pro jaký typ dat je metoda vhodná?

Analytických metod je poměrně velké množství, proto jsou tyto metody dále rozděleny do těchto skupin:

- Symbolické metody
- Metody induktivního logického programování
- Subsymbolické metody
- Metody založené na analogii v datech
- Statistické metody

V následujících kapitolách si představíme rozhodovací stromy a asociační pravidla, které patří do skupiny symbolických metod a neuronové sítě, patřící do skupiny metod subsymbolických. [1]

2. Rozhodovací stromy

Model rozhodovacího stromu je vyjádřen jako řada jednoduchých rozhodovacích pravidel, často prezentovaná ve formě grafu. Indukce rozhodovacích stromů patří k nejznámějším algoritmům z oblasti symbolických metod, které se využívají i v jiných oborech, než v dataminingu, např. klíče k určování rostlin a živočichů v biologii. Díky své snadné interpretaci si rozhodovací stromy získaly značnou popularitu. Tyto grafy mohou být snadno, bez hlubokých znalostí statistických metod, interpretovány řídícími pracovníky. [4]

2.1 Algoritmus TDIDT

Algoritmus TDIDT (Top Down Induction of Decision Trees) je základní algoritmus pro tvorbu rozhodovacího stromu. Při tvorbě rozhodovacího stromu se postupuje metodou „rozděl a panuj“ (divide and conquer). Trénovací data se postupně rozdělují do menších a menších podmnožin (tzv. uzly stromu) tak, aby v těchto podmnožinách převládaly příklady jedné třídy. Postupuje se metodou shora dolů, počínaje stromem s jedním uzlem (kořenem). Algoritmus TDIDT se provádí ve třech krocích:

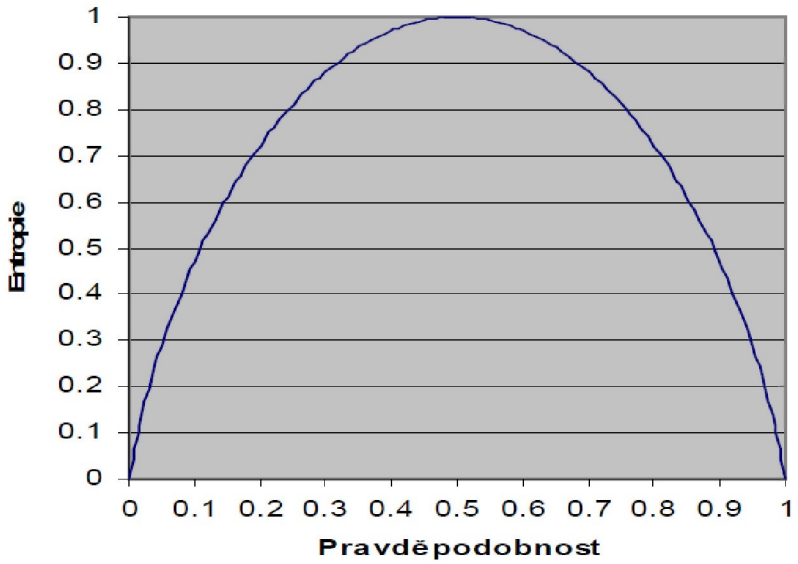
1. Zvolit jeden atribut jako kořen dílčího stromu.
2. Rozdělit data v tomto uzlu podle hodnot zvoleného atributu a přidat uzel pro každou podmnožinu.
3. Pokud existuje uzel, pro který nepatří všechna data do téže třídy, pro tento uzel opakovat body 1. a 2., jinak skončit.

Klíčovou otázkou algoritmu je vybrat vhodný atribut pro větvení stromu. Nejlepší možnost výběru je atribut, který nejlépe odliší příklady různých tříd. V Clementine (starší název pro SPSS PASW Modeler) jsou následující termíny kritériem pro větvení rozhodovacích stromů: *entropie*, *informační zisk, poměrný informační zisk,* χ^2 *(Chí kvadrát test)*, nebo *Gini index.* [10]

Základní algoritmus rozhodovacího stromu využívá jako kritérium pro větvení entropii. **Entropie** je jedním ze základních pojmů v přírodních vědách (fyzika, teorie informace…). Vyjadřuje míru neuspořádanosti nějakého systému. V teorii informace je entropie definována jako funkce

$$H = -\sum_{t=1}^{T}(p_t * \log_2 p_t)$$

kde p_t je pravděpodobnost výskytu třídy t a T je počet tříd. Průběh funkce entropie v případě dvou tříd vypadá tak, že je-li p = 1 (všechny příklady patří do této třídy) nebo p=0 (žádný příklad nepatří do této třídy), je entropie nulová. Jsou-li obě třídy zastoupeny stejným počtem příkladů (p=0,5), je entropie maximální (obr. 2.1). [1]



Obr. 2.1 Graf entropie (převzato z [])

Mějme atribut $\boldsymbol{A}$, který je použit v testu a má $\boldsymbol{v}$ různých hodnot.

1) Pro každou hodnotu v spočítáme entropii $H(A(v))$ na skupině příkladů, které jsou pokryty kategorií $A(v)$

$$H(A(v)) = -\sum_{t=1}^{T}\frac{n_t(A(v))}{n(A(v))}\log\frac{n_t(A(v))}{n(A(v))}$$

2) Dále spočítáme střední entropii $H(A)$ jako vážený součet entropií $H(A(v))$. Váhy v součtu jsou relativní četnosti kategorií $A(v)$ v datové množině. Vzorec pro výpočet střední entropie je následující

$$H(A) = \sum_{v \in Val(A)} \frac{\mathrm{n(A(v))}}{\mathrm{n}} \mathrm{H(A(v))}$$

3) Pro větvení stromu vybereme atribut s nejmenší entropií *H(A)*. [1]

2.2 Prořezávání stromů

Rozhodovací stromy, vzniklé automaticky výše uvedeným algoritmem, nemusí mít optimální tvar, vhodný pro rychlou a korektní klasifikaci. Proto jsou stromy modifikovány tzv. *ořezáváním stromů* (tree-pruning). Jsou používány dva způsoby ořezávání stromů:

- **Ořezávání při konstrukci** (pre-pruning)
- **Ořezávání po konstrukci** (post-pruning)

Ořezávání při konstrukci rozhodovacího stromu je realizováno předčasným ukončením některých větví rozhodovacího stromu. Důvodem k ukončení konstrukce větví může být např. dostatečně velká pravděpodobnost, že data v dané větvi patří do určité třídy. Problémem je určení této hranice tak, aby rozhodovací strom nebyl příliš mělký (vysoká chybovost procesu klasifikace), ani příliš hluboký (časová náročnost klasifikace).

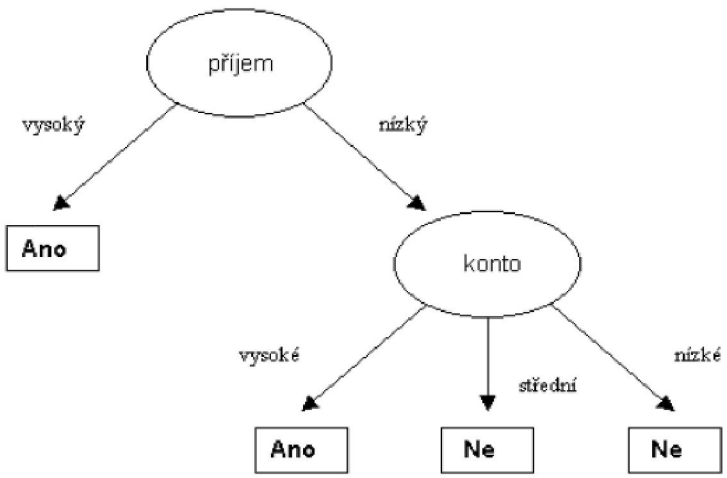
Ořezávání po konstrukci odstraňuje některé větve hotového rozhodovacího stromu. Odstraněné větve jsou nahrazeny listy, které jsou ohodnoceny třídou s největší četností v listech podstromu dané odstraněné větve. Větve jsou ořezávány, dokud je míra chyby menší, než určitá hranice. Ořezávání po konstrukci stromu je více výpočetně náročné, než ořezávání při konstrukci, ale obecně poskytuje lepší výsledky.[9]

Při vytváření redukovaného stromu je klíčovou otázkou, jak poznat, kdy lze nelistový uzel nahradit listem. Jednou z možností je použít nová (validační) data pro testování uvažované redukce. Druhou možností je odhadnout vhodnost redukce použitím statistického testu z trénovacích dat. Následující algoritmus odhaduje vhodnost prořezávání na základě trénovacích dat. Příkladem prořezávání budiž prořezávání založené na těchto pravidlech:

1. Převeď strom na rozhodovací pravidla

2. Generalizuj pravidlo odstranění podmínky za předpokladu, že dojde ke zlepšení odhadované správnosti
3. Uspořádej prořezaná pravidla podle odhalované správnosti; v tomto pořadí budou pravidla použita pro klasifikaci
4. Spočítej správnost pravidla na trénovacích datech jako podíl správně klasifikovaných příkladů pokrytých pravidlem a všech příkladů pokrytých pravidlem
5. Spočítej směrodatnou odchylku této správnosti
6. Vezmi dolní odhad správnosti pro zvolený interval spolehlivosti jako hledanou charakteristiku pravidla

Tedy například pro interval spolehlivosti 95% bude dolní odhad správnosti pravidla pro nová data = *správnost na trénovacích datech – 1,96 směrodatná odchylka.* Za použití uvedeného postupu získáme tento prořezaný strom na obr. 2.2 [1]



Obr. 2.2 Prořezaný strom (převzato z [10])

2.3 Práce s numerickými atributy

Pokud při tvorbě rozhodovacích stromů používáme numerické atributy, musíme řešit problém s velkým počtem možných hodnot. Nelze tedy vytvářet samostatnou větev pro každou hodnotu. Pomocí se jeví rozdělení oboru hodnot na intervaly. Tyto intervaly považujeme za diskrétní hodnoty atributu. Systém Clementine má metodu diskretizace zabudovanou v jednotlivých modelech rozhodovacích stromů. [10]

Nejjednodušší případ rozdělení se nazývá biorizace, tj. rozdělení na dva intervaly. Základem je využití informace o tom, do které třídy patří příklad s konkrétní hodnotou diskretizovaného atributu. Je zapotřebí najít dělící bod (cut-point), který rozdělí hodnoty do dvou intervalů. K nalezení této hodnoty musíme provést následující kroky:

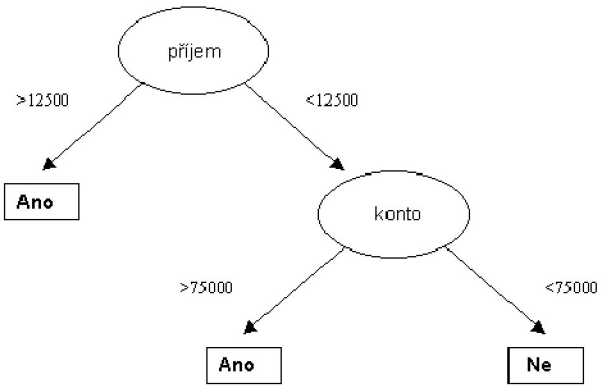
1. Seřadíme vzestupně hodnoty diskretizovaného atributu ***A***
2. Pro každou střední hodnotu dělícího bodu ***D*** spočítáme střední entropii atributu

$$H(A_D) = \frac{n(A(<D))}{n} H(A(<D)) + \frac{n(A(>D))}{n} H(A(>D))$$

První člen součtu se týká příkladů, které mají hodnotu atributu menší než ***D***. Člen ***H(A(<D)*** je hodnota entropie těchto příkladů. Druhý člen je samozřejmě pro příklady s hodnotami atributu větší než ***D***.

3. Vybereme dělící bod s nejmenší entropií. [1]

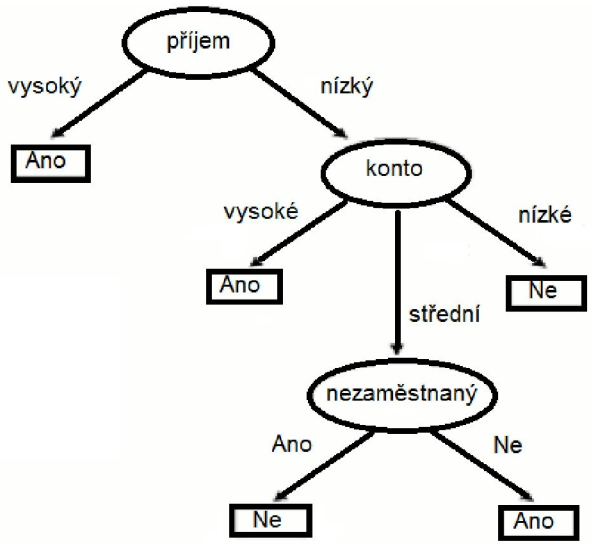
Uvedený postup binarizace se provádí on-line v průběhu vytváření stromu. V druhém kroku algoritmu TDIDT se tedy berou do úvahy jak atributy kategoriální, tak numerické. U numerických se ale nejdříve hledá vhodný práh pro binarizaci. Na rozdíl od kategoriálních atributů se může v jedné větvi stromu opakovat test na tentýž numerický atribut. Příklad rozhodovacího stromu pro numerická data je zobrazen na obr. 2.3 [1]



Obr. 2.3 Rozhodovací strom pro numerická data (převzato z [10])

2.4 Převod stromu na rozhodovací pravidla

Rozhodovací stromy lze převést na sadu rozhodovacích pravidel. Každé cestě stromem od kořene k listu odpovídá jedno pravidlo. Atributy se objeví v předpokladu pravidla, cíl bude v závěru pravidla. Převod si názorně ukážeme na příkladu rozhodovacího stromu pro možnost udělení finančního úvěru na základě příjmů. Tento strom je zobrazen na obr. 2.4 [1]



Obr. 2.4 Příklad rozhodovacího stromu (převzato z [10])

Pravidla pro tento strom budou vypadat následovně:

IF příjem (vysoký)	**THEN**	úvěr (ano)	
IF příjem (nízký) ∧ konto(vysoké)	**THEN**	úvěr(ano)	
IF příjem (nízký) ∧ konto(střední) ∧ nezaměstnaný (ano)		**THEN**	úvěr(ne)
IF příjem (nízký) ∧ konto(střední) ∧ nezaměstnaný (ne)		**THEN**	úvěr(ano)
IF příjem (nízký) ∧ konto(nízké)	**THEN**	úvěr(ne)	

2.5 Modely rozhodovacích stromů v Clementine

Clementine obsahuje čtyři různé algoritmy rozhodovacích stromů: C5.0, CHAID, QUEST a C&R Tree (klasifikační a regresní strom). Všechny tyto algoritmy vytváří rozhodovací stromy tak, že rekurzivně rozdělují data do podskupin podle predikčního pole a podle jejich vztahu k výsledku. Rozdíly modelů si popíšeme v následující podkapitole.

2.5.1 Porovnání modelů

Modely rozhodovacích stromů v Clementine jsou od sebe v některých aspektech odlišné. Rozdíly modelů z pohledu uživatele jsou tyto:

Typ výstupu: C5.0 a QUEST povolují pouze symbolické výstupní pole (podle typu množiny nebo uspořádané množiny, ačkoli pořadí bude ignorováno), zatímco C&R Tree a CHAID podporují oba, symbolický a numerický výstup. Příklad z praxe: Všechny čtyři modely mohou být použity pro model úvěrového rizika, který se skládá ze tří skupin, ale pouze CHAID a C&R Tree modely mohou být použity pro vytvoření modelu předpovědi výdajů pro příští rok (např. v dolarech) pro nedávno získané zákazníky.

Typ rozdělení: Když se datová množina rekurzivně rozděluje do podskupin, C&R Tree a QUEST podporují pouze binární dělení (rozdělení do dvou skupin), zatímco CHAID a C5.0 podporují rozdělení do více než dvou skupin.

Kritéria pro výběr prediktoru: Algoritmy se liší v kritériu použitém pro rozdělování. Pro C5.0 je hlavním kritériem informační zisk a poměrný informační zisk. Když C&R Tree předpovídá symbolický výstup, používá rozptyl měření (ve výchozím nastavení Gini index [2]). CHAID používá Chí kvadrát test. QUEST používá chí kvadrát test pro kategorické predikátory a analýzu rozptylu pro průběžné výstupy.

Manipulace chybějících hodnot prediktoru[3]: Všechny algoritmy povolují chybějící hodnoty v predikačních polích, ale používají jiné metody. C5.0 používá metodu, která projde nepatrnou část každé větve stromu z uzlu, jehož rozdělení je založeno na poli s chybějícím záznamem. C&R Tree a QUEST používají náhradu predikčního pole tam, kde je potřeba projít přes záznam s chybějící hodnotou během tréninku (tj. tvorbě stromu na množině trénovacích dat). CHAID umožňuje chybějícím hodnotám být použity při tvorbě stromu a vytvoří pro ně oddělenou kategorii.

Interaktivní tvorby stromu: Tři algoritmy CHAID, QUEST a C&R Tree podporují interaktivní tvorbu stromu, úroveň po úrovni, včetně výběru specifických proměnných pro rozdělení.

---

[2] číselná charakteristika diverzifikace (rozrůznění)
[3] ukazatel úspěšnosti a průběhu procesu či odhadce dobré i špatné prognózy

Růst velkých stromů a prořezávání: Jak se rozhodovací stromy zvětšují a houstnou, snižuje se procento případů, které projdou stromem jakoukoli cestou. Takovéto husté stromy se za 1) nemusí dobře zobecnit na data a za 2) mohou mít pravidla, která lze aplikovat pouze na malou část dat. Algoritmy C5.0, QUEST a C&R Tree nechávají růst velké stromy, a poté je prořezávají. Rozdíl je ovšem v kritériích průřezu. C5.0 obsahuje navíc volby pro přesnost (maximální přesnost na tréninkovém modelu) nebo všeobecnost (výsledky se lépe zobecňují pro zbylá data). Všechny čtyři algoritmy umožňují kontrolovat minimální velikosti podskupin (jejich definice se liší jen mírně), to pomáhá zamezit růstu větví na několika záznamech.

Množiny pravidel: Všechny algoritmy lze prezentovat jako množinu pravidel pro symbolický výstup. Množiny pravidel mohou být lépe interpretovatelné, než komplexní rozhodovací stromy. Nicméně rozhodovací strom produkuje unikátní klasifikaci pro každý datový záznam, zatímco může na záznam aplikovat jedno, nebo víc pravidel. To dodává rozhodovacímu stromu komplexnost. [12]

2.6 Model C5.0

C5.0, někdy označován též jako See5, je nejběžnější model rozhodovacího stromu zastoupený v drtivé většině data miningových systémů. Jak již bylo zmíněno v porovnávání modelů, algoritmus C5.0 využívá jako kritérium větvení tzv. *informační zisk (information gain)* a *poměrný informační zisk (information gain ratio).*[10]

Informační zisk i poměrný informační zisk jsou míry odvozené z entropie. **Informační zisk** se spočítá jako rozdíl entropie pro celá data (pro cílový atribut) a pro uvažovaný atribut. Měří tak redukci entropie způsobenou výběrem atributu A:

$$\text{Zisk}(A) = H(C) - H(A)$$

kde

$$H(C) = -\sum_{t=1}^{T} \left(\frac{n_t}{n} \log \frac{n_t}{n}\right)$$

Na rozdíl od entropie, kde jsme hledali atribut s minimální hodnotou, u informačního zisku nás zajímá atribut s maximální hodnotou. To je logické, uvážíme-li, že entropie počítané

pro celá data nezávisí na atributu. První člen rozdílu je konstantní. Rozdíl tedy bude maximální, pokud druhý člen rozdílu bude minimální. [1]

Nevýhodou informačního zisku je fakt, že se nebere v úvahu počet hodnot zvoleného atributu. V úvahu se bere pouze to, jak dobře tento atribut od sebe odliší příklady různých tříd. Pokud bychom zvolili jako atribut větvení např. pořadové číslo příkladu, dosáhneme nejnižší (nulové) entropie, protože jedné hodnotě atributu odpovídá jediný objekt. Na základě tohoto atributu bychom tedy mohli bezchybně klasifikovat trénovací data, ovšem tento atribut je zcela nepoužitelný pro klasifikaci nových příkladů. [10]

Z tohoto důvodu se někdy používá jako kritérium **poměrný informační zisk**, který bere do úvahy i počet hodnot atributu.

$$Poměrný\_zisk(A) = \frac{Zisk(A)}{Větvení(A)}$$

Větvení(A) je v tomto vztahu vlastně entropie dat vzhledem k hodnotám atributu A [4]:

$$Větvení(A) = -\sum_{v \in Val(A)} \left(\frac{n(A(v))}{n} \log \frac{n(A(v))}{n} \right)$$

V případě entropie jsme hledali atribut s minimální hodnotou, u informačního zisku je to přesně naopak.

V každém uzlu stromu algoritmus C5.0 vybírá z dat jeden atribut, který nejvíce efektivně rozdělí množinu vzorků na podmnožiny. Každá podmnožina má svoji třídu. Neexistují dvě podmnožiny v jedné třídě. Kritériem těchto tříd je rozdíl v entropii (normalizovaný informační zisk), který je výsledkem výběru atributu pro rozdělení dat. Atribut s největším normalizovaným informačním ziskem je vybrán pro rozhodování. C5.0 algoritmus pak vrací menší podseznam. [12]

[4] V předchozím případě se počítala entropie vzhledem k cílovým třídám.

Algoritmus pracuje v několika základních krocích:

- Pokud všechny vzorky v seznamu patří do stejné skupiny, vytvoří listový uzel rozhodovacího stromu a doporučí výběr této třídy
- Žádná z vlastností neposkytuje žádný informační zisk. C5.0 vytvoří rozhodovací uzel umístěný výše ve stromu s použitím očekávané hodnoty třídy.
- Objeví se instance dříve neviděné třídy. Jako v předchozím případě C5.0 se vytvoří rozhodovací uzel umístěný výše ve stromu s použitím očekávané hodnoty třídy. [12]

2.6.1 Vlastnosti modelu C5.0

Prořezávání a třídění atributů v C5.0

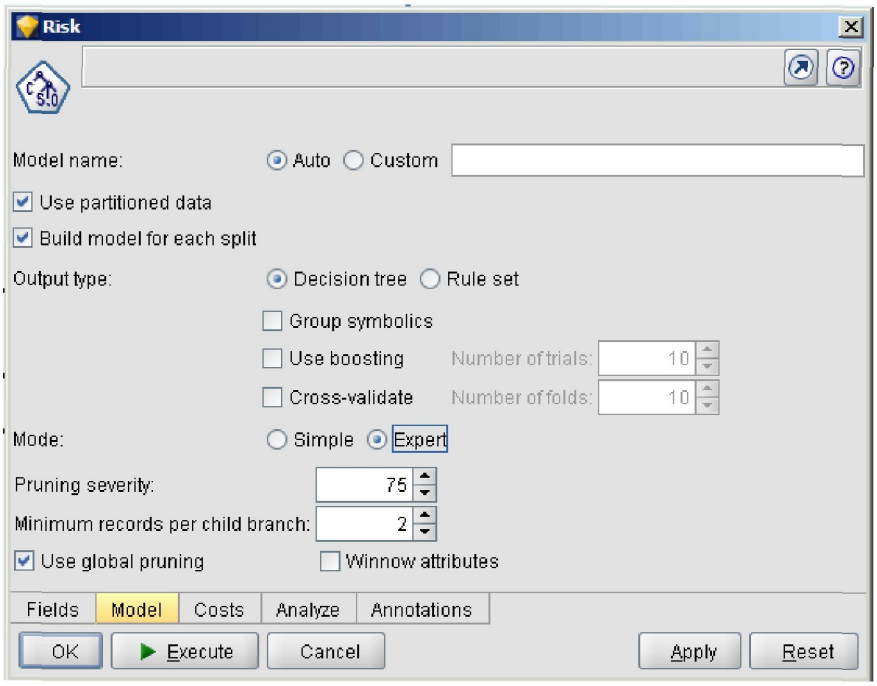
Jak již bylo řečeno C5.0 algoritmus může po sestavení stromu zpětně prořezávat větve stromu. To se dělá především kvůli tvorbě více obecného stromu (méně košatého). V expert módu uzlu C5.0 existuje volba *Prunning severity*, která nám umožňuje kontrolovat rozsah prořezávání. Čím větší číslo, tím obecnější strom vznikne jako výsledek.

V první fázi prořezávání se pracuje s podstromy, tedy s detailnějšími pohledy na strom. Na základě porovnání předpověděných chyb podstromu (tj. neprořezaných větví) a chyb v listu (prořezaného uzlu) se algoritmus rozhodne, zda by větev měla být odříznuta až k nadřazenému uzlu. Odhad chyb listů a podstromů je vypočten na základě množiny neviditelných případů stejné velikosti, jako je trénovaná množina. Vzorec pro výpočet předpokládaných chyb na listu zahrnuje počet případů v rámci listu, tedy počet těchto případů, které byly nesprávně zařazeny do tohoto listu a intervaly spolehlivosti na základě binomického rozdělení.

Druhá fáze prořezávání (*global pruning*) je posléze aplikována standardně, přičemž se bere ohled především na výkon stromu jako celku. Tato volba může být vypnuta.

Další volbou, která ovlivňuje výsledný tvar stromu, je *Winnow attributes*(v překladu „oddělení zrna od plev“). Účelem této volby je zahodit některé vstupy modelu před sestavováním rozhodovacího stromu. Tímto vyprodukujeme model, pracující s menším počtem vstupů, ovšem s téměř stejnou přesností, což může být výhodnější vzhledem

k aplikaci modelu. Této volby se využívá speciálně v případě, kdy máme hodně vstupů, které jsou statisticky příbuzné. [12]



Obr. 2.5 Volby modelu C5.0

Minimální počet záznamů na větvi

Další funkcí modelu, kterou je potřeba při stavbě základního stromu vzít v úvahu, je velikost koncového uzlu. Přesněji snažíme se zabránit tomu, aby nebyly tyto uzly příliš malé. V modelu C5.0 můžeme kontrolovat hodnotu *Minimum records per child branch*. Původní nastavená hodnota pro minimum je dva. Tato hodnota specifikuje fakt, že v každém štěpení stromu musí nejméně dva podstromy pokrýt alespoň tuto hodnotu případů. Zvýšení může být užitečné při použití datové množiny s velkým šumem a má tendenci tak produkovat méně košaté stromy. [12]

Jak používat prořezávání a minimální počet záznamů

V uzlu C5.0 existuje základní a expertní mód. Na základě výběru jednoho z nich a nastavených hodnot bude vytvořen model rozhodovacího stromu. Při použití základního módu nastavujeme pouze hodnotu *Expected noise* (velmi zřídka využívané v praxi), minimální počet záznamů na větvi je pak polovina této hodnoty. V expertním módu máme dvě možnosti:

- Pokud upřednostňujeme přesnost modelu, tak *Pruning severity*(přísnost prořezání) nastavíme na 75 a minimum záznamů na 2. I když je strom přesný, je zde stupeň obecnosti, který nepovoluje uzlům obsahovat pouze jeden záznam.
- Pokud chceme, aby výsledný model byl více obecný, *Pruning severity* je nastavena na 85 a minimum záznamů na 5. [12]

Zesílení (Boosting)

C5.0 má zvláštní způsob na zlepšení přesnosti, nazvaný zesílení. Pracuje na principu sekvenčního sestavení (sestavení jeden po druhém) mnoha modelů. První model je sestaven obvyklým způsobem. Druhý model je následně sestaven tak, že se zaměří na záznamy, které první model neklasifikoval správně. Třetí model pracuje nad chybami druhého modelu a tak dále. Nakonec jsou všechny případy roztříděny/klasifikovány aplikováním celé sady modelů s využitím vážené volící procedury, která zkombinuje jednotlivé předpovědi (výsledky modelů) do jediné souhrnné předpovědi. Zesílení může významně zvýšit přesnost modelu C5.0, zároveň ale vyžaduje delší trénink. Volba *Number of trials* umožňuje zvolit, kolik modelů bude pro výsledek metody zesílení použito.

Zesílení může na první pohled nabízet zpřesnění dat bez následků („zadarmo"), ve skutečnosti tomu tak není. Když doběhne sestavování modelů, existuje pro předpovědi více než jeden strom. Proto pak nelze jednoduše popsat výsledný model a ani to, jak jednotlivý prediktor ovlivní výsledek. To může být vážný nedostatek, a proto je *zesílení* použito jen v situacích, kdy je hlavním cílem analýzy prediktivní přesnost, ne pochopení významu modelu. [12]

Náklady chybné klasifikace (Misclassification Costs)

Tabulka *Costs tab* umožňuje nastavit náklady chybné klasifikace. Při použití stromu k predikci symbolických výstupů je někdy potřeba mít tuto hodnotu nastavenou, aby se model vyhnul „drahým chybám" při jeho zkreslení. Ovládací prvky chybné klasifikace dovolují specifikovat hodnoty pro všechny takové klasifikace. Základní hodnota je 1.0, aby byly všechny chybné klasifikace stejně nákladné. Při nastavení různých hodnot nákladů má výsledný strom sklon k tvorbě „méně drahých" chybných klasifikací, obvykle za cenu toho, že počet těchto chyb se zvýší. [12]

2.7 Model CHAID

CHAID neboli Chi-squared Automatic Interaction Detector je algoritmus, který se používá ke konstrukci nebinárního rozhodovacího stromu. Kritériem dělení stromu u modelu CHAID je **χ^2 (chí-kvadrát) test**. Vzorec pro výpočet chí-kvadrát testu vypadá následovně

$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}$$

O_i je pozorovaná (výběrová) četnost a E_i je očekávaná (teoretická) četnost. Máme nulovou hypotézu H_0, což je náš předpoklad (parametr rozložení), u kterého testujeme, zda se dá zamítnout. Provádíme rozdělení četností do intervalů. Podmínkou pro vykonání testu je, že žádný interval nemá nulovou četnost a 20% intervalů má četnost menší než 5. Výsledkem testu je „H0 zamítnuto" nebo „H_0 nezamítnuto". Pokud je hodnota testovacího kritéria vyšší, než takto klasifikovaná kritická hodnota rozdělení (chí-kvadrát pro v=(r-1) stupňů volnosti, kde r = počet intervalů), hypotézu o shodě dvou rozdělení zamítáme. [14]

Algoritmus postupuje v těchto krocích:

- **Příprava prediktorů**: CHAID automaticky seskupuje hodnoty kategoriálních atributů. Souvislé prediktory rozdělí na několik kategorií s přibližně stejným počtem pozorování (výskytů).
- **Seskupování hodnot atributu**: Zvolí se dvojice kategorií atributů, které jsou si nejpodobnější z hlediska χ^2 a které mohou být spojeny. Nová kategorizace atributu je považována v daném kroku za možné seskupení hodnot.
- **Výběr dělící proměnné:** Pomocí χ^2 testu se spočítá pravděpodobnost *p* pro každý z možných způsobů seskupení. Seskupení s nejnižší pravděpodobností *p* se zvolí jako „nejlepší". Nakonec algoritmus zjišťuje, zda toto nejlepší seskupení hodnot statisticky významně přispěje k odlišení příkladů různých tříd. [12]

2.7.1 Vlastnosti modelu CHAID

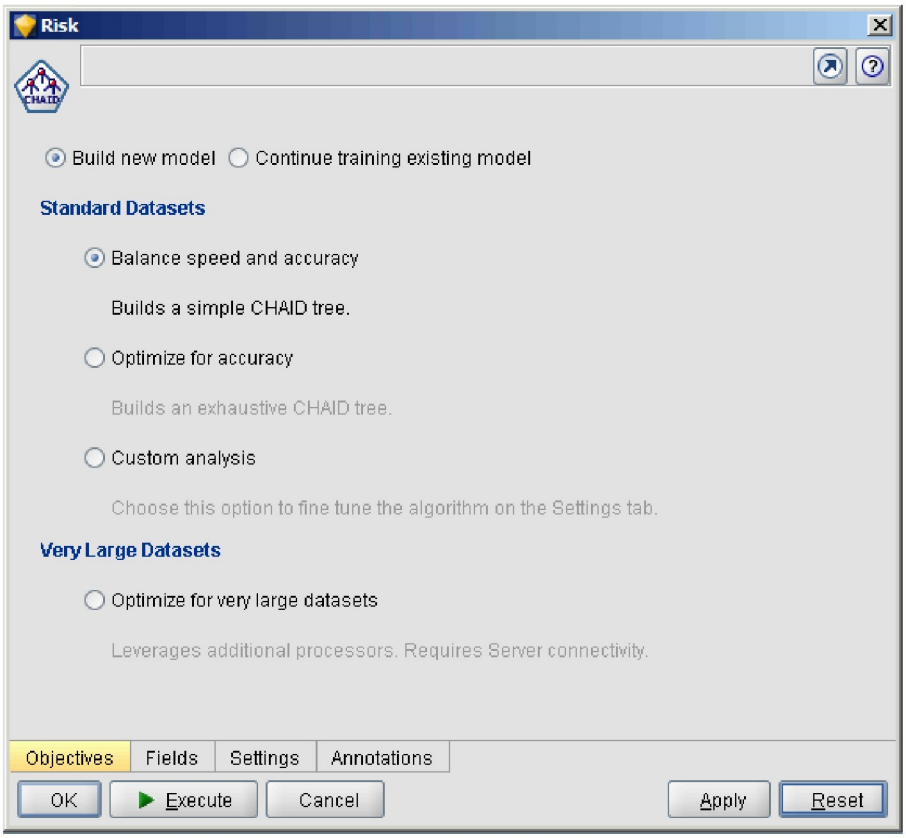
Při modelování symbolických hodnot jsou v uzlu *CHAID* k dispozici dvě metody. *CHAID* a *Exhaustive CHAID*. Smyslem této modifikace je odstranění některých slabin modelu CHAID.

Pokud v Clementine zvolíme možnost *Exhaustive CHAID*, provádí se důkladnější testování a spojení prognostických proměnných. Přesněji řečeno se krok *seskupování hodnot atributu* opakuje, dokud se nevytvoří pouze dvě skupiny hodnot atributu. Pokud chceme Exhaustive CHAID použít na velkou množinu dat s velkým počtem souvislých prediktorů, může tato operace vyžadovat značnou výpočetní dobu.

CHAID umožňuje pouze jednu možnost v nastavení modelu, maximální hloubku stromu (*maximum tree depth*). Účelem této volby je omezení velikosti modelu, protože CHAID nepoužívá funkci prořezávání. Uživatel si nastaví hloubku pomocí možnosti *Levels below root*, která je na začátku nastavena na hodnotu 5. Tato volba by měla záviset na velikosti datového souboru, počtu prediktorů a komplexnosti kýženého výsledku.

CHAID tedy vybírá prediktor dělení pomocí χ^2 testu a automaticky nastavuje význam těchto hodnot při jejich testování. Důvodem je velký počet těchto testů vykonávaných v průběhu algoritmu.

Pomocí tlačítka *Stopping* kontrolujeme zastavení růstu stromu na základě velikosti uzlu. Hodnota pro zastavení může být definována jako absolutní počet záznamů, nebo jako procento z celkového počtu záznamů. Ve výchozím nastavení je hodnota pro rodičovskou větev 2% ze záznamů; dětská větev pak nejméně 1%. Je pohodlnější pracovat spíše s celkovým počtem záznamů, než s procenty, ale v každém případě tyto hodnoty musíme nastavit, abychom dostali menší nebo větší strom. [12]



Obr. 2.6 Volby modelu CHAID

2.8 Model C&R Tree

Třetí model rozhodovacích stromů v Clementine je C&R Tree. Kritériem pro větvení stromu je u tohoto modelu *impurity („znečištění“)*. Zachycuje stupeň, do jaké míry jsou odezvy v rámci uzlu koncentrovány do jednotlivých výstupních kategorií. Tzv. čistý uzel je takový, ve kterém všechny případy zapadají do jedné kategorie, zatímco maximálně „nečistý“ uzel obsahuje stejný počet případů v každé výstupní kategorii. [12]

Základní měřítko pro výpočet znečištění je tzv. **Gini index**. Vzorec pro výpočet gini indexu vypadá následovně [1]

$$\text{Gini} = 1 - \sum_{t=1}^{T} (\text{p}_\text{t})^2$$

p_t je opět relativní počet příkladů t-té třídy na nějaké (pod)množině. Hodnotu Gini indexu pro jeden atribut spočítáme analogicky jako hodnotu entropie pro jeden atribut. Tedy tak, že pro každý atribut spočítáme vážený součet indexu pro jednotlivé hodnoty atributu, přičemž váhy se opět rovnají relativní četnosti příslušných hodnot. [1]

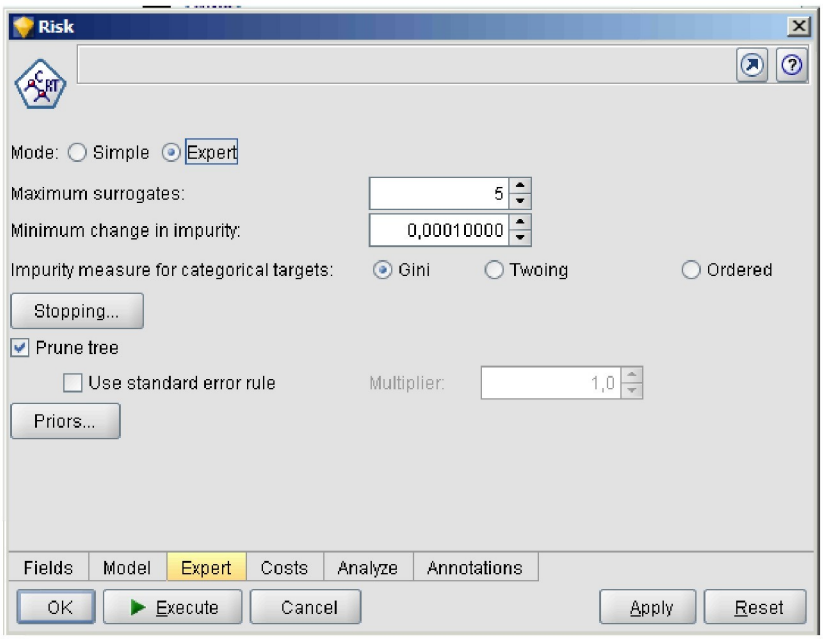
$$\mathrm{Gini(A)} = \sum_{v \in \mathrm{Val(A)}} \frac{n(A(v))}{n} \mathrm{Gini}(A(v)), \qquad \mathrm{Gini}(A(v)) = 1 - \sum_{t=1}^{T} \left(\frac{n_t(A(v))}{n(A(v))} \right)^2$$

Pro větvení se použije atribut, který bude mít nejmenší hodnotu tohoto indexu. Analogicky k informačnímu zisku entropie můžeme i zde maximalizovat i rozdíl mezi Gini indexem, počítaným pro cílový atribut, a Gini indexem jednoho atributu, tedy Gini(C) – Gini(A). Gini index cílového atributu se výpočítá tímto vzorcem [1]

$$Gini(C) = 1 - \sum_{t=1}^{T} \left(\frac{n_t}{n} \right)^2$$

2.8.1 Vlastnosti modelu C&R Tree

V jednoduchém nastavení modelu existuje pouze jediná volba, kterou se model zaobírá, a to *maximum tree depth*. Hodnota je pro začátek nastavena na 7. Při stejném nastavení bude mít výsledný model hlubší strukturu než CHAID. Vzhledem k možnosti prořezávání modelu (*pruning*) a dalších pravidel pro zastavení tvorby modelu, aktuální hloubka stromu může mít méně úrovní, než je nastaveno. [12]



Obr. 2.7 Volby modelu C&R Tree

Prořezávání C&R stromu

Zaškrtávací políčko *Prune tree* aktivuje prořezávání výsledného modelu. *Standart error rule* (standardní chybové pravidlo) umožňuje modelu vybrat nejjednodušší strom, jehož odhad rizika je nejblíže podstromu s nejmenším rizikem. *Multiplier* ukazuje, jak velká chybová odchylka je povolena mezi finálním stromem a stromem s nejmenším rizikem. Pokud hodnotu zvedneme, prořezávání stromu bude přísnější.

Tlačítko *Stopping* kontroluje zastavení růstu stromu na základě velikosti uzlu, obdobně jako u modelu CHAID. [12]

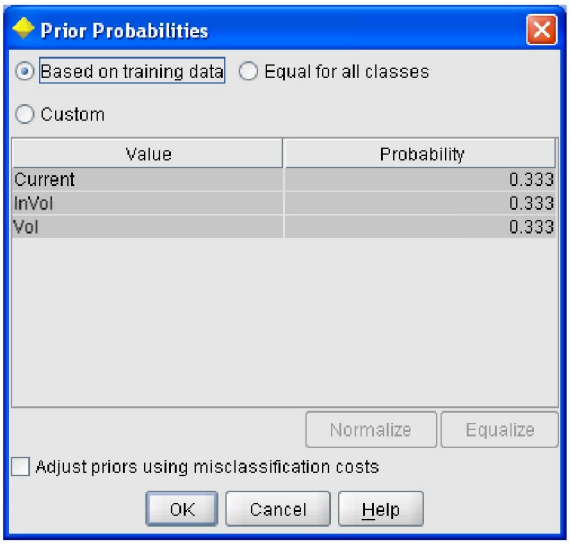
Náhrady (Surrogates)

Náhrady se používají v případech, kdy potřebujeme ošetřit chybějící hodnoty v predátorech. V každém rozdělení stromu model C&R Tree identifikuje vstupní hodnoty, které jsou nejvíce podobné vybranému rozdělenému poli. Pokud záznam určený ke klasifikaci obsahuje chybějící hodnoty pro dělení pole, vybere se v poli náhrad tohoto záznamu jiná hodnota, která může být pro rozdělení stromu použita.

Volba *Maximum surrogates* kontroluje počet náhradních polí, které budou uloženy na každém uzlu. Udržování více náhradní polí zpomaluje proces tvorby modelu, základní hodnota (5) je obvykle dostačující. [12]

Tlačítko Priors

Ve výchozím nastavení jsou předchozí rozdělení pravděpodobnosti (priors) nastaveny tak, aby souhlasily s hodnotami nalezenými v trénovacích datech. Volba *Equal for all classes* umožňuje nastavit všechny pravděpodobnosti na shodnou hodnotu. Pomocí možnosti *custom* si můžeme tyto hodnoty nastavit podle svého. Hodnoty se musí ve výsledku rovnat 1, takže pokud máme nastaveny hodnoty vzhledem k našim požadavkům, ale nerovnají se 1, tlačítko *normalize* je upraví. Pravděpodobnosti mohou být upraveny na základě nákladů chybné klasifikace (*misclassification costs*) vložených v tabulce *Costs tab.* [12]



Obr. 2.8 Předchozí rozdělení pravděpodobnosti

Náklady chybné klasifikace v C&R Tree

I v tomto modelu máme možnost nastavit hodnoty pro náklady chybné klasifikace. Nastavení se od modelu C5.0 prakticky v ničem neliší. Základní hodnoty jsou opět rovny 1, tedy náklady na chybnou klasifikaci jsou vyrovnané. [12]

2.9 Model QUEST

QUEST (Quick Unbiased Efficient Statistical Tree) je binární klasifikační metoda částečně vyvinutá pro zkrácení doby zpracování potřebné pro rozsáhlé C&R Tree analýzy s velkým počtem polí anebo záznamů, nebo s velkým počtem polí i záznamů současně. QUEST se také pokouší omezit tendenci modelu C&R Tree upřednostňovat prediktory, které umožní větší rozdělení stromu. [12]

Podobně jako CHAID a C&R Tree umožňuje i QUEST nastavení maximální hloubky stromu (v jednoduchém nastavení). Výstupem může být model nebo interaktivní strom.

QUEST od sebe odděluje operace výběru prediktoru a rozdělení v uzlu. Kritériem je podobně jako v modelu CHAID statistický test. Pro trvalé i řadové proměnné je analýza rozptylu a důležitost F-testu. Pro jmenné proměnné (typu *flag* a *set*) se provádí chí-kvadrát test. Následně je vybrán prediktor s nejmenší výslednou hodnotou F-testu nebo Chí-kvadrát testu. Tento model je efektivnější, než předchozí C&R Tree, neboť nejsou zkoumána všechna

větvení a nejsou testovány všechny kategorické kombinace při vyhodnocování výběrového prediktoru. [12]

Po výběru prediktoru se algoritmus rozhoduje, jak by mělo být pole rozděleno (do dvou skupin). K tomu využívá **kvadratickou diskriminační analýzu**, za použití vybraného prediktoru na třídy formované cílovými kategoriemi. [12]

Při diskriminační analýze předpokládáme, že ke každé třídě (hodnotě nominální veličiny) c_t, $t=1,\ldots,T$ existuje *(diskriminační) funkce* f_t taková, že

$f_t(\mathbf{x}) = \max_k f_k(\mathbf{x})$, k=1,...,T

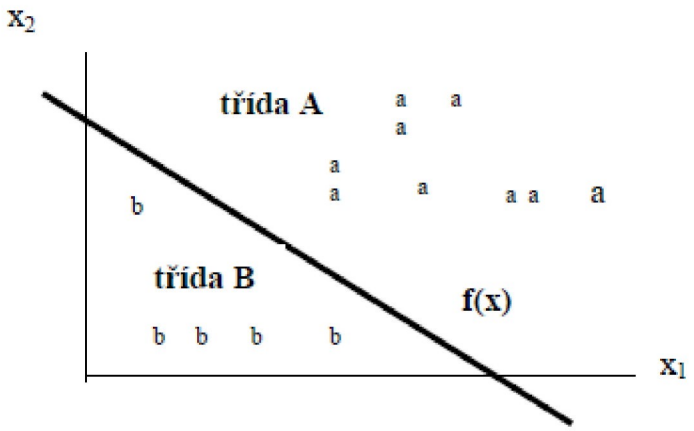
právě když příklad $\mathbf{x}=[x1, x2, \ldots, xm]$ patří do třídy c_t. V případě *lineární diskriminační analýzy* mají diskriminační funkce *ft* podobu lineární kombinace

$$f_t = q_{0\,j} + q_{1\,j}\,x_1 + q_{2\,j}\,x_2 + \ldots. + q_{m\,j}\,x_m.$$

Diskriminace do dvou tříd je nejjednodušší případ, kdy místo funkcí *f1* a *f2* můžeme hledat funkci

$$f(\mathbf{x}) = f1(\mathbf{x}) - f2(\mathbf{x}).$$

Příklady pak můžeme klasifikovat podle znaménka této funkce. Pokud si představíme příklady jako body v *m*-rozměrném prostoru veličin, bude funkce *f(**x**)* představovat nadrovinu v tomto prostoru, oddělující od sebe příklady obou tříd. Pro *m=2* je funkce *f(x)* přímka. (viz následující obrázek) [15]



Obr. 2.9 Lineární diskriminační analýza

Pokud chceme zaručit optimalitu klasifikace ve smyslu minimální chyby, musíme použít jako diskriminační funkce podmíněné (aposteriorní) pravděpodobnosti zařazení pozorování **x** do třídy c_t. [15]

$$f_t(x) = P(c_t|x) = \frac{P(x|c_t)\,P(c_t)}{\sum_k P(x|c_k)P(c_k)}$$

Ve variantě pro dvě třídy lze opět použít rozdíl aposteriorních pravděpodobností [15]

$$f(\boldsymbol{x}) = f_1(\boldsymbol{x}) - f_2(\boldsymbol{x}) = P(\boldsymbol{x}|c_1)\,P(c_1) - P(\boldsymbol{x}|c_2)\,P(c_2).$$

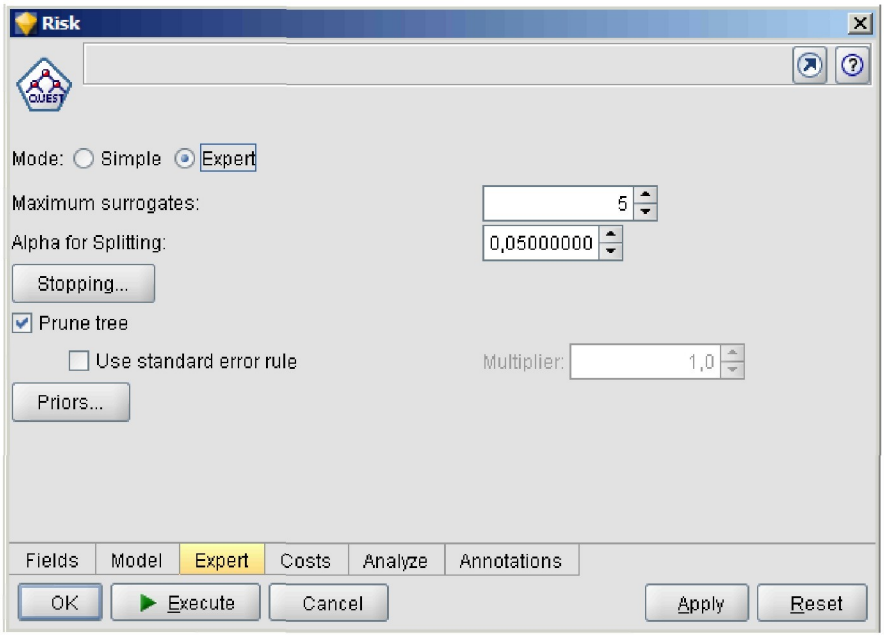
Pokud se uvedené pravděpodobnosti *P(***x***/ck) P(ck)* řídí normálním rozdělením, lze odvodit podobu kvadratické diskriminační funkce [15]

$$f(x) = \frac{1}{2}X^T(S_1^{-1} - S_2^{-1})X^T + (\mu_1^T S_1^{-1} - \mu_2^T S_2^{-1})X + \frac{1}{2}\ln\frac{|S_2|}{|S_1|} - \frac{1}{2}(\mu_1^T S^{-1}\mu_1 - \mu_2^T S^{-2}\mu_2) - ln\frac{P(c_1)}{P(c_2)}$$

V QUEST je nastavena hodnota důležitosti (*alpha for splitting*) na hodnotu 0,05 v diskriminační analýze a lze ji samozřejmě změnit. Při práci s velkými soubory se doporučuje snížit tuto hodnotu například na 0,01. [12]

2.9.1 Prořezávání, Stopping, Náhrady v QUEST

Prořezávání stromu QUEST, náhrady prediktorů pro chybějící hodnoty a funkce tlačítka *stopping* funguje stejně jako u modelu C&R Tree. [12]



Obr. 2.10 Volby modelu QUEST

3. Asociační pravidla

Metoda dolování asociačních pravidel, spojená přímo s procesem dobývání znalostí z databází, vznikala v devadesátých letech minulého století. V roce 1993 ji popsal Rakesh Agrawal. Popis metody souvisel s *analýzou nákupního košíku,* jenž tuto metodu zpopularizoval. Cílem analýzy je odhalit zvyky nakupujících a hledat závislosti mezi různým zbožím, které si zákazníci vloží do svého nákupního košíku. Znalost informace o tom, které výrobky zákazníci obvykle kupují současně, mohou v praxi pomoci v tvorbě katalogů, pomáhají při definování strategie rozmístění zboží v prodejně atd.[5] [1]

3.1 Základní tvar

Základní tvar asociačního pravidla je: ***Ant =>Suc***, kde Ant je předpoklad (antecedent) a Suc je závěr (sukcedent). U těchto pravidel vytvořených z dat nás pro **n** příkladů zajímá:

- n(Ant & Suc) = ***a***, počet objektů pokrytých předpokladem i závěrem
- n(Ant & ¬Suc) = ***b***, počet objektů pokrytých předpokladem, ale nepokrytých závěrem
- n(¬Ant & Suc) = ***c***, počet objektů nepokrytých předpokladem, ale pokrytých závěrem
- n(¬Ant & ¬Suc) = ***d***, počet objektů nepokrytých ani předpokladem, ani závěrem.

n(Ant) = a + b = r, n(¬Ant) = c + d = s, n(Suc)= a + c = k, n (¬Suc) = b + d = l,

n=a + b + c + d.

Z těchto čísel můžeme počítat různé charakteristiky pravidel a kvantitativně tak hodnotit nalezené znalosti. Místo o „počtu objektů" se někdy mluví o *četnosti,* resp. *frekvenci kombinace.*[11]

[5] Nešlo o první počin, který se v této oblasti objevil. Mnohem dříve totiž skupina českých vědců okolo P. Hájka přišla s konceptem asociačních pravidel pod názvem metoda GUHA. Tento koncept byl vytvořen již na přelomu 70. a 80. let.

Základní mírou relevance asociačních pravidel jsou *podpora* a *spolehlivost*. **Podpora** je počet objektů, který vyjadřuje podíl všech hodnot pokrytých předpokladem i závěrem (hodnota ***a***) a všech záznamů.

$$P(Ant\ \&\ Suc) = \frac{\mathrm{a}}{\mathrm{n}}$$

Spolehlivost (nazývaná též jako platnost – validity, konzistence – consistency, nebo správnost – accuracy) je podmíněná pravděpodobnost závěru, pokud platí předpoklad, tedy

$$P(Suc|Ant) = \frac{\mathrm{a}}{a+b}$$

Další důležité charakteristiky asociačních pravidel jsou:

1. absolutní, popř. relativní, počet objektů, které splňují předpoklad:

$a + b$, tedy

$$P(Ant) = \frac{\mathrm{a}+\mathrm{b}}{\mathrm{n}}$$

2. absolutní, popř. relativní, počet objektů, které splňují závěr:

a + c, tedy

$$P(Suc) = \frac{\mathrm{a}+\mathrm{c}}{\mathrm{n}}$$

3. pokrytí (coverage), neboli podmíněná pravděpodobnost předpokladu pokud platí závěr:

$$P(Suc|Ant) = \frac{\mathrm{a}}{\mathrm{a}+\mathrm{c}}$$

4. kvalita, tj. vážený součet spolehlivosti a pokrytí:

$$Kvalita = \mathrm{w}_1 \frac{a}{a+b} + w_2 \frac{a}{a+c},$$

kde w_1 a w_2 se obvykle volí tak, aby se součet w_1 a w_2 rovnal 1. [1]

Dalšími charakteristikami asociačních pravidel mohou být:

5. kauzální podpora

$$P(Ant\ \&\ Suc) + P(\neg Ant\ \&\ \neg Suc) = \frac{\mathrm{a} + \mathrm{d}}{\mathrm{n}}$$

6. kauzální spolehlivost

$$\frac{1}{2}P(Suc \mid Ant) + \frac{1}{2}P(\neg Ant \mid \neg Suc) = \frac{1}{2}\frac{\mathrm{a}}{\mathrm{a} + \mathrm{b}} + \frac{1}{2}\frac{d}{b + d}$$

7. deskriptivní potvrzení

$$P(Ant\ \&\ Suc) - P(Ant\ \&\ \neg Suc) = \frac{\mathrm{a} - \mathrm{d}}{\mathrm{n}}$$

8. kauzální potvrzení

$$P(Ant\ \&\ Suc) + P(\neg Ant\ \&\ \neg Suc) - 2P(Ant\ \&\ \neg Suc) = \frac{\mathrm{a} + \mathrm{d} - 2\mathrm{b}}{\mathrm{n}}$$

9. ujištění

$$\frac{P(Ant)P(\neg Suc)}{P(Ant\ \&\neg Suc)} = \frac{(a + b)(b + d)}{dn}$$

10. zajímavost

$$\frac{P(Ant\ \&\ Suc)}{P(Ant)P(Suc)} = \frac{an}{(a + b)(a + c)}$$

11. závislost

$$P(Suc|Ant) - P(Suc) = \frac{a}{a+b} - \frac{a+c}{n} \quad [11]$$

Všechny tyto charakteristiky asociačních pravidel jsou vytvořeny z kontingenční tabulky, viz tabulka 1. Další vzorce počítané z této tabulky by tentokrát již vyjadřovaly různé typy pravidel.

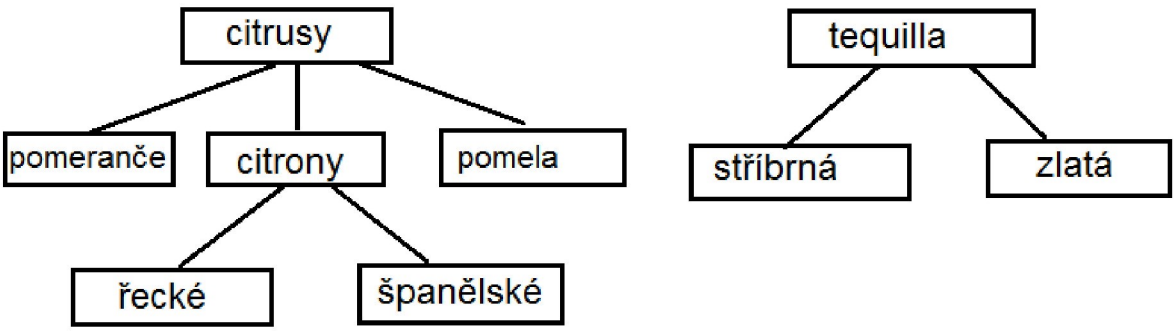
Tabulka 1. Kontingenční tabulka (bez chybějících hodnot)

	Suc	!Suc	∑
Ant	**A**	**b**	r
!Ant	**C**	**d**	s
∑	K	l	**n**

3.2 Zobecněná asociační pravidla

Princip zobecněných asociačních pravidel si nejlépe ukážeme na výše zmíněné *analýze nákupního košíku.* Obchod nabízí různé druhy zboží. Zboží, které si zákazník uloží do košíku je součástí přirozené taxonomie. Principem analýzy nákupního košíku je rozmístění dvou položek po obchodě co možná nejdál od sebe, aby zákazník při cestě za druhou surovinou potkal a zakoupil další produkt.

Můžeme zavést množinu všech položek $\mathbf{I} = \{i_1,\ldots,i_n\}$ a jednotlivé transakce $\mathbf{T_k}$ (pro k=1,…,m) chápat jako podmnožinu množiny **I**. Zobecněná pravidlo je tedy ve tvaru **A => B,** kde $\mathbf{A} \cap \mathbf{B} = \theta$. Příkladem nám budiž tequilla a citrony. Zajímají nás tedy pravidla na nejnižší úrovni hierarchie, např. Stříbrná tequilla => citrony(Řecko), ale i obecnější pravidla, využívající této taxonomie tequilla => citrony. [1]



Obr. 3.1 Taxonomie sortimentu zboží

Problém při hledání zobecněných pravidel je ve vhodné volbě minimální požadované *podpory* a *spolehlivosti*. **Podpora *p*** je pravděpodobnost výskytu množiny položek $\mathbf{A} \cup \mathbf{B}$ (tedy všech položek obsažených v **A** i **B**) v množině transakcí (datové tabulce). **Spolehlivost *s*** je pravděpodobnost výskytu **B** v transakcích, které obsahují **A.** Tedy:

$p = \text{podpora}(A \rightarrow B) = P(A \cup B)$, $s = \text{spolehlivost}(A \rightarrow B) = P(B|A)$ [9]

V následujících tabulkách jsou uvedeny údaje o transakcích. Tabulka 2 obsahuje seznam položek. Tabulka 3 ukazuje položky s podporou alespoň 50%. Tabulka 4 pak ukazuje pravidla s podporou alespoň 50 % a spolehlivostí alespoň 60%.

Tabulka 2. Nákupy

nákup	položky
1	pomelo
2	citrony (Španělsko)
3	citrony (Řecko), stříbrná tequilla
4	pomaranče (Řecko), zlatá tequilla

Tabulka 3. Četnosti

položka	četnost
pomeranče (Řecko)	2
tequilla	2
citrony	3
citrusy	4

Tabulka 4. Zobecněná asociační pravidla

Pravidlo	Podpora (%)	Spolehlivost (%)
Citrony => tequilla	50	66
Tequilla => citrony	50	100
Tequilla => citrusy	50	100

Množina položek M se nazývá **frekventovaná**, jestliže dosahuje minimální stanovené podpory P_p, tedy $P(M) \geq p_p$. [9]

Asociační pravidlo A => B se nazývá **silné**, jestliže množina položek $A \cup B$ je frekventovaná, $(P(A \cup B) \geq p_p)$ a spolehlivost pravidla dosahuje minimální stanovené hodnoty s_p $(P(B|A) \geq s_p)$. Je tedy jasné, že výsledkem algoritmu hledání by měl být seznam všech silných asociačních pravidel. [9]

3.3 Algoritmy asociačních pravidel v Clementine

Clementine obsahuje tři různé uzly, které vyhledávají v datech asociační pravidla, Apriori, GRI a Carma. První dva zmíněné přistupují k hledání obdobně. Oba Apriori i GRI začínají generováním jednoduchých pravidel, která testují oproti datové množině. Nejvíce efektivní pravidla ze všech vytvořených, ta která splňují specifická kritéria spolehlivosti a podpory a dalších vyhodnocovacích měřítek, jsou uchována. Následně jsou tato pravidla rozšířena o další podmínky z třetího pole (tento proces se nazývá specializace) a opět jsou

testována oproti datům. Nejzajímavější pravidla jsou uchována a pokračuje specializace dokud žádné pravidlo nesplňuje minimální kritéria, nebo dokud nebylo dosaženo maximálního počtu předpokladů.

Naproti tomu Carma (zkratka pro Continuous association rule mining algorithm), používá pouze dva postupy práce v datech. V prvním identifikuje často využívané množiny položek, které mají tendenci se vyskytovat současně. Následně v druhé fázi vypočítává přesné frekvence těchto kandidátských množin a ty, které splní kritéria podpory a spolehlivosti, jsou zachovány. [12]

3.3.1 Model Apriori

Nejznámějším algoritmem pro hledání asociačních pravidel je algoritmus apriori. Tento algoritmus pracuje pouze se symbolickými daty, tedy typu *flag* a *sets*. Jako takový je mnohem efektivnější a rychlejší než GRI algoritmus (viz. níže) při použití stejně velké množiny dat. Další vlastnost, která výrazně zrychluje práci tohoto algoritmu, je fakt, že apriori využívá chytré indexování dat a minimalizuje tak nutnost procházet kompletní datovou množinu při generování asociačních pravidel. [12]

Jádrem tohoto algoritmu je hledání často se opakujících množin položek (frequent itemsets). Jedná se o kombinace kategorií, které dosahují předem zadané četnosti v datech. Algoritmus probíhá v následujících krocích: [1]

1. do L_1 přiřaď všechny kategorie, které dosahují alespoň požadované četnosti
2. polož k = 2
3. dokud $L_{k-1} \neq \emptyset$
    a. pomocí funkce apriori-gen vygeneruj na základě L_{k-1} množinu kandidátů C_k
    b. do L_k zařaď ty kombinace z C_k, které dosáhly alespoň požadovanou četnost
    c. zvětši počítadlo k

Funkce apriori-gen(L_{k-1})

1. pro všechny dvojice kombinací $Comb_p$, $Comb_q$ z $L_{k,-1}$

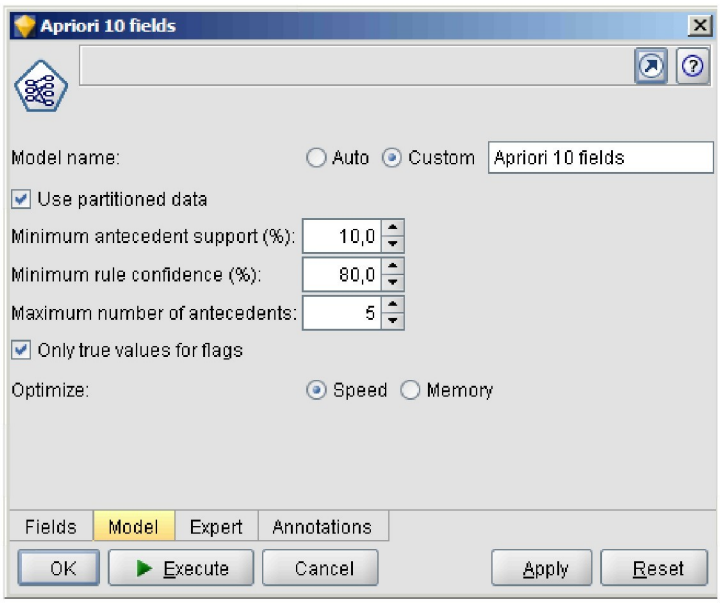
pokud se $Comb_p$ a $Comb_q$ shodují v k-2 kategoriích, přidej $Comb_p \wedge Comb_q$ do C_k

2. pro každou kombinaci Comb z C_k

pokud některá z jejich podkombinací délky k-1 není obsažena v L_{k-1}, odstraň Comb z C_k

Po nalezení kombinací, které vyhovují svojí četností, se vytvářejí asociační pravidla. Každá kombinace Comb se rozdělí na všechny možné dvojice podkombinací *Ant* a *Suc* takové, že *Suc =Comb – Ant*. Uvažované pravidlo *Ant => Suc* má pak podporu, která se rovná četnosti kombinace *Comb*. Spolehlivost pravidla se spočítá jako podíl četností kombinací *Comb* a *Ant*. Při vytváření pravidel se využívá skutečnosti, že četnost nadkombinace je nejvýše rovna četnosti kombinace $n(Comb1) \geq n(Comb2)$. [11]

Apriori má tři kritéria výběru pravidel: minimální podpora, minimální spolehlivost pravidla a maximum předpokladů pravidla. Základní hodnoty jsou nastaveny na 10% podpory, 80% spolehlivosti, a 5 předpokladů. Takto nastavené hodnoty pravděpodobně nebudou odpovídat žádné částečné datové množině. Podpora je snížena typicky hned na začátku analýzy, aby bylo možné vygenerovat více potencionálních pravidel, ačkoliv to částečně závisí na individuální prioritě a základním hodnocení položek. Všimněte si, že minimální podpora nebo minimální spolehlivost je nastavena na příliš vysokou hodnotu pro částečné datové množiny a nebudou tedy vygenerována žádná pravidla. [12]



Obr. 3.2 Volby modelu Apriori

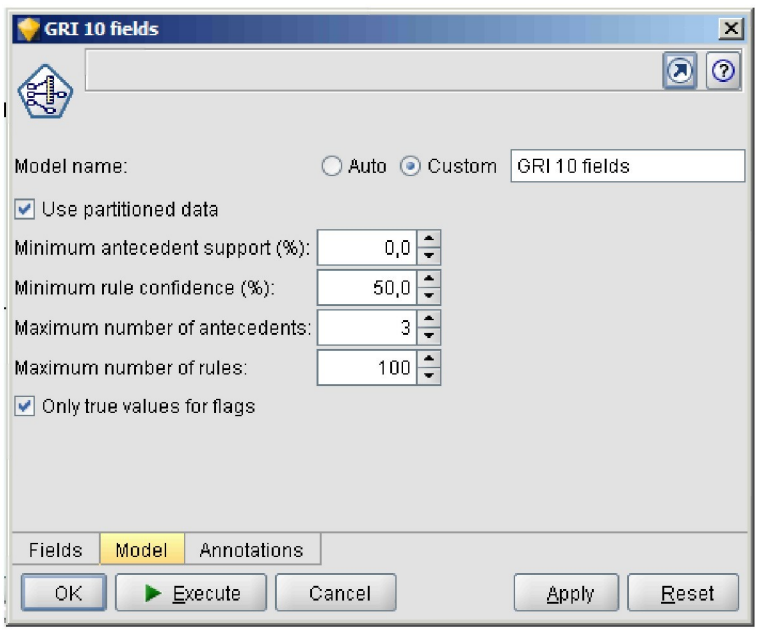
3.3.2 GRI (Generalized Rule Induction)

Algoritmus GRI může být použit pro širší datovou množinu, než Apriori, a používá rozdílná měřítka k rozhodnutí, jaké pravidlo je zajímavé. GRI umí používat numerická pole na vstupech (jako předpoklady) pravidel, ovšem pouze symbolická (flagy, sety) pro vyvození závěrů.

GRI generuje asociace založené na informačním obsahu pravidla. Informační obsah nebo též „zajímavost" (viz výše) je v uzlu GRI označen jako *J measure*, které porovnává podporu a spolehlivost. Použitím tohoto měřítka se nemusí vyhledat všechna potencionální pravidla hned po generování první množiny pravidel složených z všech možných jednoduchých pravidel.

Základní hodnota pro podporu je u GRI 0%, což znamená, že zde není žádná počáteční hodnota pro podporu pravidla, na rozdíl od Apriori. A základní hodnota minima spolehlivost je 50%, tedy nižší než 80% u modelu Apriori.

Z počátku je hodnota maxima předběžných podmínek nastavena na tři a maximum generovaných pravidel je nastaven na 20. Všechny tyto rozdíly znamenají, že i když oba uzly používají stejný algoritmus, je pravděpodobné, že najdou poněkud jiné výsledky v množině pravidel. [12]

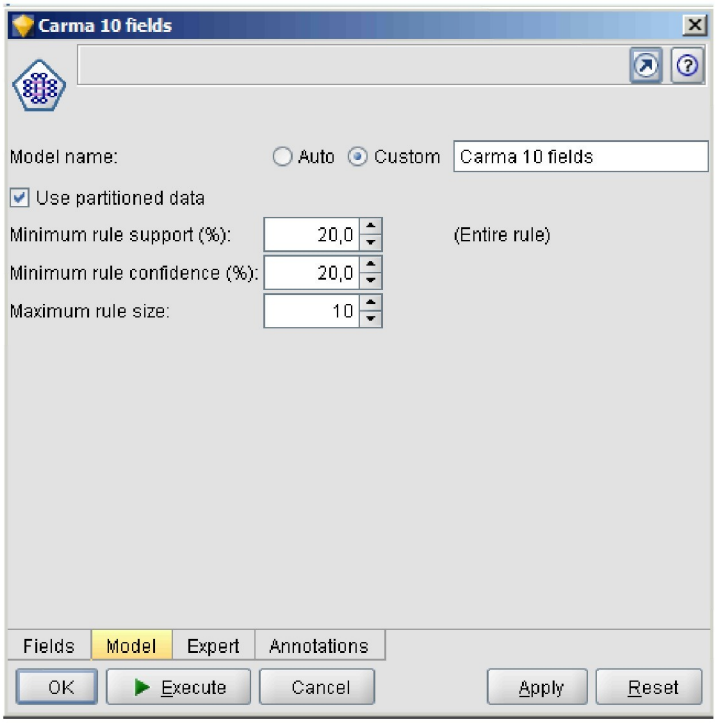


Obr. 3.3 Volby modelu GRI

3.3.3 Carma

Tento algoritmus využívá pouze symbolických proměnných typu *flag* při načítání standardních tabulkových dat (jeden záznam = jeden případ). Pole mohou mít jakýkoli směr (v uzlu type). Pokud nastavíme všechna pole na *both*, je model Carma ekvivalentní k modelu Apriori (také se všemi poli nastavenými na oba směry, vstupní i výstupní). Omezení, které položky budou předpoklady a závěry, můžeme nastavit po sestavení modelu.

Stejně jako předchozí algoritmy i Carma používá tři parametry k tvorbě asociačních pravidel: podporu a spolehlivost pravidla a maximální velikost pravidla. Tyto hodnoty jsou ze začátku nastaveny na 20% podpory, 20% spolehlivosti, a 10 předpokladů v asociačním pravidle a jsou pravidelně změněny uživatelem. Obvykle se při prvním kroku snižuje podpora pravidla. [12]



Obr. 3.4 Volby modelu Carma

3.4 Generování kombinací

Generování kombinací (konjunkcí) hodnot atributů je základem všech algoritmů hledání asociačních pravidel. Při generování prohledáváme prostor všech přípustných konjunkcí, v kterých se nesmí opakovat atributy. Metod prohledávání je několik:

- *Do šířky* – Kombinace se generují tak, že se nejprve vygenerují všechny kombinace délky jedna, pak všechny kombinace délky dvě atd.
- *Do hloubky* – Začíná se od první kombinace délky jedna, a ta se pak prodlužuje (vždy o první kategorii dalšího atributu), pokud nelze přidat další kategorii atributu, změní se kategorie posledního atributu, pokud nelze ani to, pak se kombinace zkrátí a zároveň se změní poslední kategorie.
- *Heuristická* – Generuje kombinace podle četností v pořadí od nejvyšší četnosti k nulové. [1]

Příklad generování kombinací:

Tabulka 5. Data pro generování kombinací [11]

Klient	Příjem	Konto	Pohlaví	Nezaměstnaný	úvěr
K1	Vysoký	Vysoké	Žena	Ne	Ano
K2	Vysoký	Vysoké	Muž	Ne	Ano
K3	Nízký	Nízké	Muž	Ne	Ne
K4	Nízký	Vysoké	Žena	Ano	Ano
K5	Nízký	Vysoké	Muž	Ano	Ano
K6	Nízký	Nízké	Žena	Ano	Ne
K7	Vysoký	Nízké	Muž	Ne	Ano
K8	Vysoký	Nízké	Žena	Ano	Ano
K9	Nízký	Střední	Muž	Ano	Ne
K10	Vysoký	Střední	Žena	Ne	Ano
K11	Nízký	Střední	Žena	Ano	Ne
K12	Nízký	Střední	Muž	Ne	Ano

Tabulka 6. Generování do šířky

Č.	četnost	kombinace
1	7,00	1n
2	5,00	1v
3	4,00	2n
4	4,00	2s
5	4,00	2v
6	6,00	3m
7	6,00	3z
8	6,00	4a
9	6,00	4n
10	8,00	5a
11	4,00	5n
12	2,00	1n2n
13	3,00	1n2s
14	2,00	1n2v
15	4,00	1n3m
16	3,00	1n3z
.	.	.
323	0,00	1v2v3z4n5n

Tabulka 7. Generování do hloubky [11]

Č.	četnost	kombinace
1	7,00	1n
2	2,00	1n 2n
3	1,00	1n 2n 3m
4	0,00	1n 2n 3m 4a
5	0,00	1n 2n 3m 4a 5a
6	0,00	1n 2n 3m 4a 5n
7	1,00	1n 2n 3m 4n
8	0,00	1n 2n 3m 4n 5a
9	1,00	1n 2n 3m 4n 5n
10	0,00	1n 2n 3m 5a
11	1,00	1n 2n 3m 5n
12	1,00	1n 2n 3z
.	.	.
.	.	.
.	.	.
.	.	.
.	.	.
323	4,00	5n

Tabulka 8. Generování podle četností [11]

Č.	Četnost	kombinace
1	8,00	5a
2	7,00	1n
3	6,00	3m
4	6,00	3z
5	6,00	4a
6	6,00	4n
7	5,00	1v
8	5,00	1n 4a
9	5,00	4n 5a
10	5,00	1v 5a
.	.	.
.	.	.
.	.	.
.	.	.
203	1,00	1v 2s 3z 4n 5a
.	.	.
323	0,00	1v 2n 3z 4n 5n

S růstem počtu atributů roste značně i počet kombinací. Při odhadu počtu kombinací se musíme ptát, „jak velký je prostor kombinací, který se prohledává?“. Označme K_{A1}, $K_{A2},\ldots,K_{An}$ počet kategorií atributu $A_1,A_2,\ldots,A_n$ (n je počet atributů, ze kterých vytváříme kombinace).

Pak počet kombinací délky jedna je

$$\sum_{i=1}^{n} K_{A_i},$$

délky dvě

$$\sum_{j=1,i=1,i\neq j}^{n} K_{A_i} K_{A_j},$$

až počet všech kombinací

$$\prod_{j=1}^{n}(1 - A_j) - 1.$$

Tyto výpočty se týkají počtu všech možných kombinací, bez ohledu na řídící parametry generování a na vstupní data. Pokud požadujeme pouze kombinace do určité délky, tak počet generovaných kombinací samozřejmě klesne. V tabulce 7 je vidět, že pro data v tabulce 5 je 323 syntakticky správných kombinací, ale s nenulovou četností jich je pouze 203. [1]

3.5 Speciální případy asociačních pravidel

3.5.1 Pravidla s výjimkami

Výsledkem algoritmů pro hledání asociačních pravidel bývá rozsáhlý seznam pravidel, která je nutno interpretovat. Vodítkem je obvykle nějakým způsobem definovaná zajímavost. Za zajímavá pravidla se doporučuje považovat ta pravidla, která se vymykají běžným představám (tzv. common sence). Pravidla s výjimkami se definují pomocí následující trojice pravidel:

- Pravidlo odpovídající ustáleným představám – $Comb_A \Rightarrow Suc$

  např. *použité bezpečnostní pásy* $\Rightarrow$ *přežití automobilové havárie*

- Pravidlo s hledanou výjimkou – $Comb_A \wedge Comb_B \Rightarrow \neg Suc$

  např. *použité bezpečnostní pásy* $\wedge$ *předškolní věk* $\Rightarrow$ *úmrtí při havárii*

- Referenční pravidlo – $Comb_B \Rightarrow \neg Suc$

např. *předškolní věk⇒ úmrtí při havárii* je zřejmé, že toto pravidlo má nízkou podporu, nebo spolehlivost, případně obojí [1]

Takovéto trojice můžeme hledat až ve výsledném souboru pravidel. Prohledávání při generování pravidel je řešeno simultánní hledání dvojice obecného pravidla a výjimky ($Comb_A$, $Comb_B$) v prostoru všech dvojic kombinací. Prohledávání probíhá „do hloubky" a je řízeno parametry $P_{S_1}, P_{S_2}, P_{F_1}, P_{F_2}, P_{I_2}$, které specifikují kvantitativní charakteristiky nalezených pravidel:

$$P(Comb_A) \geq P_{S_1},$$

$$P(Suc|CombA) \geq P_{F_1},$$

$$P(CombA \wedge CombB) \geq P_{S_2},$$

$$P(\neg Suc|\ CombA \wedge CombB) \geq P_{F_2},$$

$$P(\neg Suc|CombB) \leq P_{I_2}.$$

Pokud nesplní nějaká dvojice ($Comb_A$, $Comb_B$) některý z uvedených požadavků, pak ho nesplní ani žádná z jejích nadkombinací. Prohledávání uzlu nadkombinací ($Comb_{A'}$, $Comb_{B'}$) se tedy ukončí. [1]

3.5.2 Asociace v časových sekvencích

Asociační pravidla se většinou hledají v databázích, kde se nepočítá s faktorem času. V případě, že potřebujeme nalézt asociace mezi událostmi, které se odehrávají v různých časových okamžicích, musíme prohledávat tato data a hledat opakující se *epizody* mezi těmito událostmi. Příkladem databáze s časovými údaji může být databáze poruch v telekomunikační síti. Příkladem sekvence událostí může být:

(P, 123), (Q, 125), (S, 140), (P, 150), (R, 151),

(Q, 155), (S, 201), (P, 220), (S, 222), (Q, 225). [11]

V zápise *(písmeno, číslo)* je písmeno název události a číslo je údaj o čase, ve kterém událost nastala.

Epizoda je částečně uspořádaná množina událostí a dělí se na *sériovou* a *paralelní*. **Sériová epizoda** je taková, při které na sebe různé události navazují. **Paralelní epizoda** je pak taková, kdy různé události nastanou současně. Při analýze časových sekvencí jde o to, nalézt epizody, které se opakují dostatečně často.[11]

Pracujeme-li s konkrétními časovými okamžiky, je pro hledání epizod podstatné zadání časového okna, uvnitř kterého se musí epizoda vyskytnout. Toto okno pevné délky W_D se posouvá po sekvenci událostí. Okno lze posouvat s krokem pevné délky (je-li délka časové sekvence T a krok rovný jedné, je počet všech oken, která musíme prozkoumat rovný $T-W_D+1$), nebo tak, že nové okno začíná s další událostí. Je-li například sekvencí událostí výše uvedený příklad a je-li $W_D=20$, budeme pro druhý způsob posouvání okna zkoumat okna

[P Q S], [Q S], [S P R Q], [P R Q], [R, Q], [Q], [S P], [P S Q].

V těchto oknech tedy budeme hledat četné epizody. Opět vycházíme ze skutečnosti, že má-li pro okno dané délky dostatečnou četnost epizoda $P \rightarrow Q \rightarrow R$, mají dostatečnou četnost i epizody $P \rightarrow Q$, $Q \rightarrow R$ a $P \rightarrow R$. Můžeme tedy delší epizody skládat z kratších úseků (tzv. podepizod).[1]

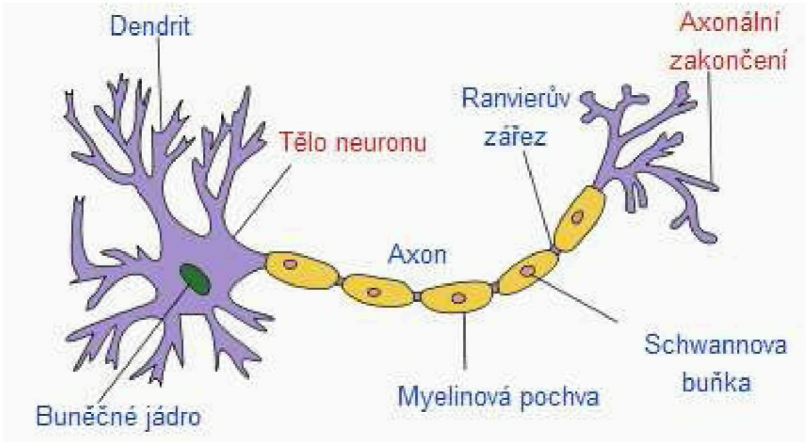
4. Neuronové sítě

Cílem této kapitoly není do detailu popsat každý aspekt neuronových sítí, ale pouze krátce přiblížit jejich funkcionalitu a modely, které jsou k dispozici v Clementine. Hlavní nevýhodou těchto modelů je, že se špatně interpretují. Na rozdíl od rozhodovacích stromů.

Neuronová síť je dalším z výpočetních modelů v BI. Její vzorem je chování odpovídajících biologických struktur, resp. mozku. Mozek je složitější, než cokoliv jiného, čemu člověk porozuměl. Je složen z živých buněk, určených ke zpracování informace, ale není navržen jen k tomu. Buňky musí mít řadu dalších vlastností, aby zůstaly naživu. Stejně jako je mozek složen z mnoha neuronů, je i neuronová síť složena z mnoha umělých neuronů, které jsou navzájem propojeny. Pro lepší představu umělého neuronu si nejdříve popíšeme ten mozkový. [16]

4.1 Neuron

Biologický neuron je základní funkční a histologická jednotka nervové tkáně. Skládá se z těla (soma) ze kterého vybíhá řada (desítky až stovky) kratších výběžků, zvaných dendrity, a jeden dlouhý výběžek (axon). Tělo neuronu má velikost v řádu mikrometrů, dendrity mají velikost několik milimetrů. Axon může dosahovat délky několika desítek centimetrů až jednoho metru. Zatímco dendrity jsou dostředivé, axon je odstředivý. Konec axonu tvoří synapse, která dosedá na jiný neuron. Přes synapse se přenášejí speciální signály (vzruchy), které neuron přenáší a zpracovává, a tím podmiňuje schopnost organismu na ně reagovat. V důsledku chemických reakcí se v místě, kde synapse dosedá na neuron, mění propustnost buněčné membrány neuronu a mění se membránový potenciál na základě kladných a záporných iontů vně a uvnitř buňky. Některé synaptické vazby snižují potenciál a některé ho zvyšují. Dílčí účinky těchto vazeb se kumulují a ve chvíli, kdy celkový membránový potenciál překročí určitý práh, je neuron aktivován, a přes svoji synapsi začne působit na další neurony, na které je napojen. Zjednodušeně řečeno umělý neuron tak přijímá kladné a záporné podněty a sám se aktivuje ve chvíli, kdy souhrn těchto podnětů překročí určitý práh. [17]

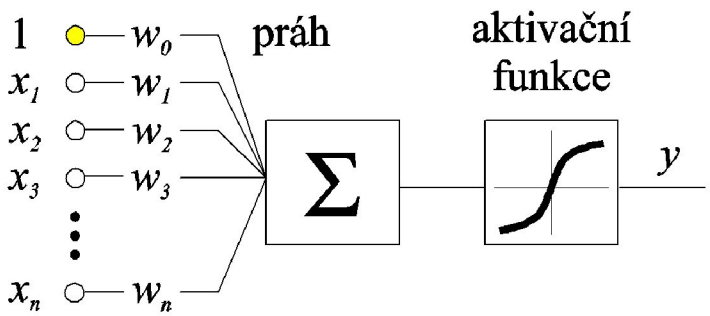


4.1 Schéma neuronu (převzato z [17])

4.2 Modely neuronů

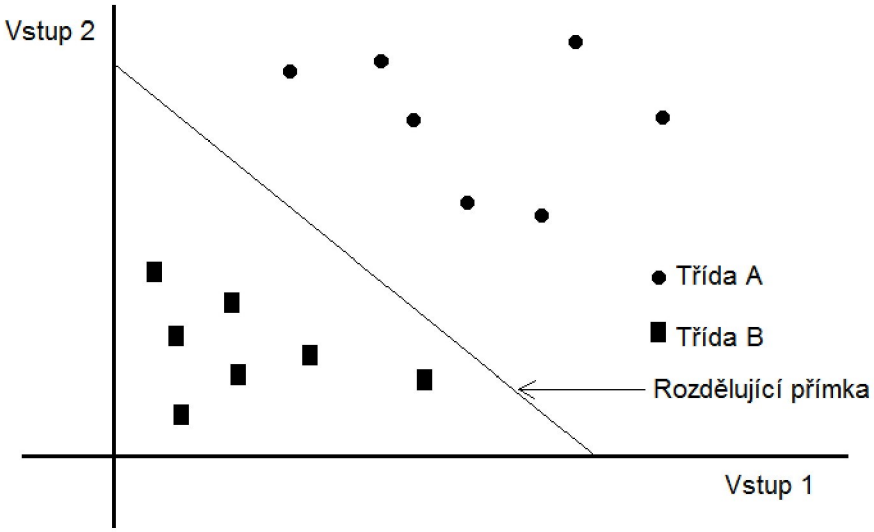
Je popsána celá řada modelů neuronu. Od těch velmi jednoduchých, používajících výstupní hodnotu nelineárních transformací souhrnu podnětů, až po velmi složité, popisující každý detail chování neuronu. Prvním modelem byl „logický neuron“, popsaný McCullochem a Pittsem v roce 1943, který pracoval pouze s binárními vstupními a výstupními hodnotami. Dalším modelem ja ADALINE (Adaptive Linear Neuron). Jedná se o jednovrstvý model neuronové sítě, kterou vynalezl profesor Bernard Windrow v roce 1960. [18]

Základem tohoto modelu je logický neuron McCullocha a Pittse. Skládá se z váhy, tendence a sčítací funkce. Vstupem do ADALINE jsou numerické hodnoty (podněty) označené $x_1, x_2, \ldots, x_m$. Každý podnět x_i je násoben vahou w_i, tento součin vstupuje do součtového členu, kde je vytvořen vážený součet $\sum_{i=1}^{m} w_i x_i$. V případě, že tento vážený součet přesáhne práh w_0, je výstup z modelu Adaline roven 1, v opačném případě 0. Na rozdíl od logického neuronu jsou vstupy libovolná čísla, výstupy zůstávají dvouhodnotové. [16]



Obr. 4.2 Schéma umělého neuronu (převzato z [18])

Činnost modelu Adaline se dá jednoduše geometricky interpretovat. Jednotlivé vstupní podněty $x_1,x_2,\ldots,x_m$ mohou představovat hodnoty vstupních atributů nějakého objektu (např. teplota, výška, váha aj.). Každý takový objekt lze pak reprezentovat jako bod v m-rozměrném prostoru. Adaline je *lineární klasifikátor*, který rozděluje objekty do dvou tříd. To znamená, že se body v prostoru vyskytují v jednom ze dvou částí prostoru, který rozděluje přímka. Pro více tříd (vícerozměrný prostor) je to pak rozdělující rovina, která dělí prostor objektů. [1]



Obr. 4.3 Lineární prostor s rozdělující přímkou (převzato z [16])

4.3 Učení neuronu

Schopnost učení neuronu je důležitá vlastnost. V mozku učení neuronů funguje tak, že se posilují ty vazby neuronu, které způsobují jeho aktivaci. Učení u umělých neuronů je algoritmus nastavení vah $\boldsymbol{w}$ na základě předložených příkladů $[x_i, y_i]$ tak, aby systém co nejsprávněji zpracovával i neznámé případy x_k.

Z představy učení mozkových neuronů vznikl v roce 1949 Hebbův zákon, který se v umělých neuronech formuluje takto:

$$W_{i+1} = w_i + y_i x_i$$

$[x_i, y_i]$ je trénovací příklad, ve kterém x_i je vektor hodnot vstupních atributů. y_i je informace o zařazení příkladu x_i do třídy. w_i je váha před modifikací a w_{i+1} je váha po modifikaci. Váha w může růst nade všechny meze, což neodpovídá biologické realitě. [1]

4.4 Rozdělení predikčních modelů

Učení neuronových sítí rozdělujeme do dvou kategorií:

- Učení s učitelem - **Supervizované**
- Učení bez učitele - **Nesupervizované**

Modely **supervizované** (s učitelem) se učí na základě získávání zkušeností ze známých příkladů z pravidla historických událostí. Pro tyto modely je potřeba přesná znalost požadované odpovědi sítě tzn., že pro tyto modely máme k dispozici kompletní učební data s příklady, která obsahují vstupy i výstupy. Po skončení procesu učení může následovat „přezkoušení" modelu. K tomuto přezkoušení jsou využity různé evaluační postupy.

Modely **nesupervizované** (bez učitele) se učí prohledáváním dat. Modely zkoumají tato data a hledají v nich skryté podobnosti a vztahy. Vstupní data obsahují pouze vstupní hodnoty. Vlastnosti cílových proměnných nejsou předem známé. Výstupem z modelu jsou nalezené vztahy, predikce podobnosti a anomální vzory.

4.5 Modely neuronových sítí

4.5.1 Perceptron

První model neuronové sítě pochází z roku 1957 a jmenuje se *Perceptron*. Jeho tvůrcem byl Frank Rosenblatt. Sestrojil první počítač Mark 1, který byl schopný se učit pomocí metody „pokus, omyl". Využíval systému založeného na modelu neuronové sítě Perceptron, který byl navržen jako model zrakové soustavy. Je to hierarchický systém tvořený třemi vrstvami.

První z nich (tzv. sítnice) má za úkol přijímání informace z prostředí. Tvoří ji prvky (receptory), které nabývají hodnot 1 nebo 0 podle toho, zda jsou prostředím excitovány nebo ne.

Druhá vrstva jsou takzvané asociativní elementy. Výstupy receptorů jsou přivedeny na tyto elementy náhodně zvolenými vazbami. Asociativní element připomíná výše zmíněný adptivní lineární neuron (adaline) s tím, že všechny váhy w_i mají pevné hodnoty +1 nebo -1.

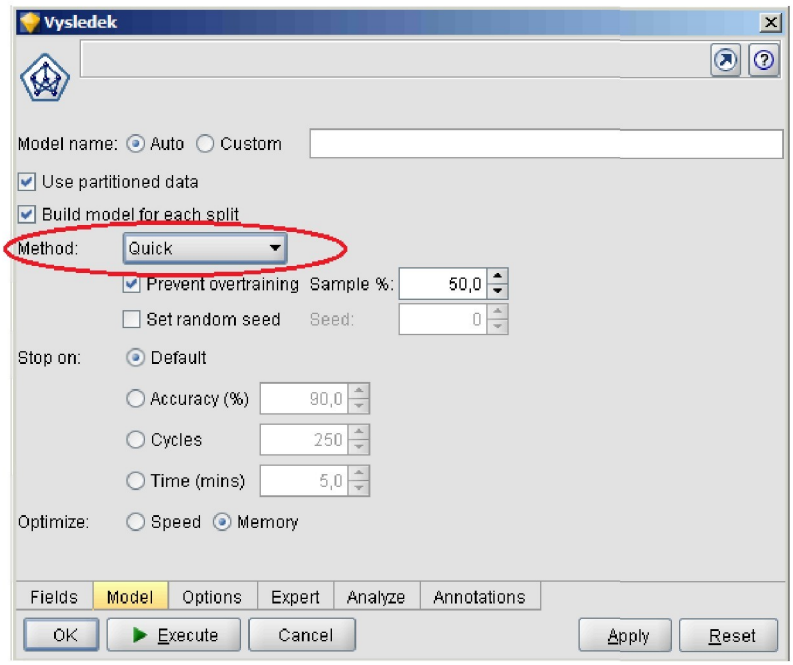
Pokud souhrn vstupů překročí zadaný práh, tak se element aktivuje (vydá hodnotu 1). V této vrstvě je asociativních elementů řádově desítky tisíc.

Výstupy z vrstvy asociativních elementů jsou náhodně zvolenými vazbami propojeny na třetí vrstvu tzv. reagujících elementů. V této vrstvě se vypočítává vážený součet $\sum_{i=1}^{m} w_i x_i$.

Výsledky tohoto kroku se vyhodnocují v posledním bloku, kde se vybere reagující element s nejvyšším výstupem pro daný obraz. Ten tedy odpovídá třídě, do které je obraz zařazen. [1]

4.6 Modely neuronových sítí v Clementine

Je popsáno mnoho různých druhů supervizovaných neuronových sítí. Ačkoliv ve světě dataminingu se nejčastěji používají pouze dva, a to jsou MLP (Multi-Layer Perceptron) a RBF (Radial Basis Function). Oba tyto modely, a různé typy jejich algoritmů, jsou obsaženy v uzlu *Neural net*. Výběr příslušného výsledného modelu provedeme v nabídce *method* (viz obr. 4.4) [12]

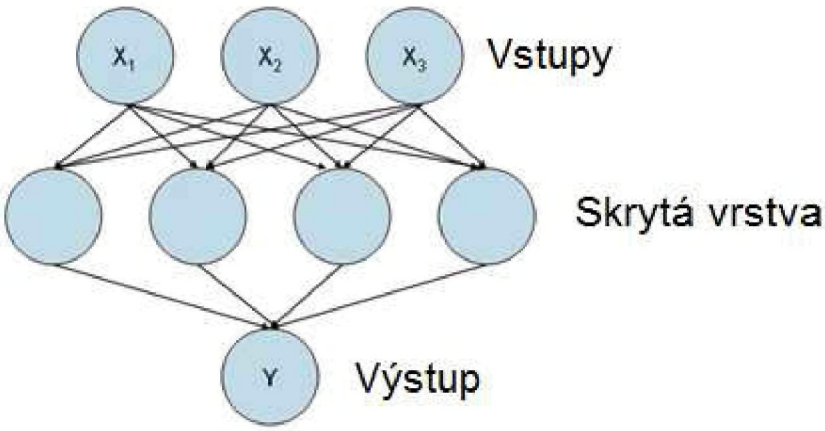


Obr. 4.4 Volby modelu neuronová síť

4.6.1 Multi-Layer Perceptron

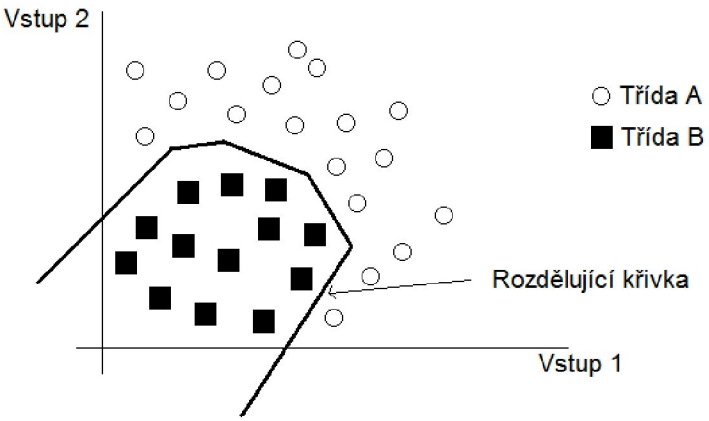
Multi-layer perceptron je v současnosti nejpopulárnější typ neuronové sítě. MLP síť je sestavena z vrstev neuronů, které jsou spojeny se všemi neurony v sousední vrstvě, ale

s neurony ve stejné vrstvě spojeny nejsou. Vazby mezi neurony mají různé váhy. Všechny MLP sítě obsahují vstupní vrstvu, výstupní vrstvu a alespoň jednu skrytou vrstvu. Skrytá vrstva je vyžadována k provedení nelineárních výpočtů. Počet neuronů je přímo úměrný komplexnosti problému. Vícevrstvá topologie je sice proveditelná, ovšem v praxi je zřídkakdy potřeba použít více než jednu skrytou vrstvu. [12]



Obr. 4.5 Jednovrstvý perceptron

Pro vizualizaci toho, jak MLP pracuje, si představme problém, ve kterém potřebujeme predikovat výsledné pole sestávající z dvou skupin za využití pouze dvou vstupních polí. Následující obrázek ukazuje graf dvou vstupních polí zakreslených vedle sebe. MLP rozdělí vstupy křivkou na základě výpočtu nelineární kombinace těchto vstupů a tím je rozdělí do dvou tříd.



Obr. 4.6 Rozhodující rovina s využitím MLP (převzato z [16])

Výhody použití MLP:

- Efektivní model pro velký rozsah možných problémů

- Schopný dobře generalizovat data
- Pokud data nejsou sdružena ve smyslu vstupních dat, model si sám klasifikuje vzory v extrémních oblastech
- V současnosti je to nejběžnější neuronová síť – je k dispozici spousta materiálů a literatury, která zkoumá její použití [12]

Nevýhody MLP:

- Může zabrat velké množství času k vytrénování modelu
- Nezaručuje nalezení nejlepšího souhrnného řešení [12]

Algoritmy MLP

Uzel *Neural Net* nabízí na výběr čtyři možnosti učících se algoritmů: Quick, Dynamic, Multiple a Prune. Navíc k možnosti Prune existuje volba Exhaustive Prune, neboli ořezání modelu s velkou přednastavenou topologií. Výběrem vhodného algoritmu volíme poměr „přesnost/doba", tedy se rozhodujeme mezi přesností a výpočetní dobou algoritmu. [12]

Quick

Volba Quick vytvoří síť, obsahující jednu skrytou vrstvu. Počet neuronů ve skryté vrstvě se určí podle několika faktorů, týkajících se počtu a typu polí použitých v analýze. Uživatel vybere, zda se mají použít až tři skryté vrstvy a může si nastavit počet neuronů v každé vrstvě. Základní nastavená hodnota *persistence* je 200 a může být změněna dle potřeby. Mohou být změněny i hodnoty *alpha* a *eta*, což jsou hodnoty míry učícího procesu. [12]

Dynamic

Dynamický trénink modelu využívá „dynamicky rostoucí" síť, která začíná se dvěma skrytými vrstvami a dvěma neurony v obou těchto vrstvách. Roste přidáním jednoho neuronu do každé vrstvy. Síť monitoruje trénink a kontroluje tzv. „přetrénování (over-training); nedostatečné zlepšení, které spouští velký růst sítě. Model ukončí svůj růst v okamžiku, kdy přidaný neuron neposkytne žádný přínos pro počet rostoucích pokusů. Tato volba je často pomalá, ale poskytuje dobré výsledky. [12]

Multiple

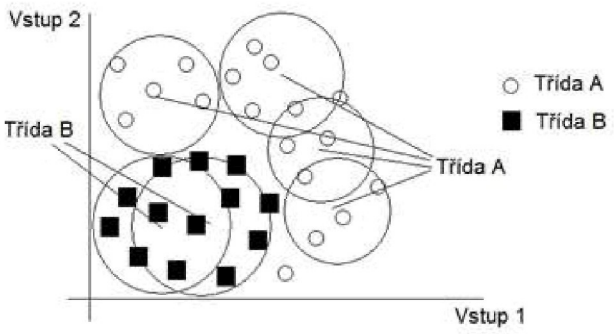
Tato metoda vytváří několik sítí. Každá z nich má různou topologii. Některé obsahují jednu skrytou vrstvu, některé dvě a všechny mají různé počty skrytých neuronů. Sítě jsou pseudo-paralelně trénovány, což způsobuje, že je tato metoda extrémně pomalá. Stejně jako předchozí ovšem může dosahovat dobrých výsledků. [12]

Prune

Metoda Prune začína s jednou nebo dvěma velkými skrytými vrstvami. Trénink je zpočátku podobný jako u metody Quick. Na skryté neurony je posléze aplikována analýza citlivosti. Nejslabší skryté neurony (na základě *Hidden rate* – standardně 15%) jsou pak ze sítě odstraněny. Tento proces trénování a odstraňování se opakuje, dokud neskončí zlepšování v síti. Když je tento proces hotov, stejný „osud" čeká také vstupní neurony (*Input Rate* je také defaultně 15%). Na konci tohoto procesu může vzniknout síť s méně predikčními poli, než měla původní. Tento algoritmus je považován za nejlepší, ovšem je časově velmi náročný. [12]

4.6.2 Radial Basis Function (RBF)

Tento model neuronové sítě je novější. Tuto síť více zajímají lokální regiony v prostoru vstupních polí. Ukážeme si grafickou reprezentaci RBF na stejném typu příkladu popsaném u předchozího modelu. Představme si, že RBF vytváří shlukování uvnitř plochy vstupních hodnot a zakroužkuje jednotlivé shluky dat podle počtu provedených bázových funkcí. Neuron, který reaguje nejsilněji, je ten, na jehož bázovou funkci ukazují vstupní data. Myšlenka RBF je jednoduchá, ale výběr středu každé bázové funkce je na druhou stranu poměrně obtížný. [12]



Obr. 4.7 Operace RBF (převzato z [16])

Výhody RBF:

- Je rychlejší v trénování modelu, než MLP
- Umí modelovat data shlukovaná v prostoru vstupů [12]

Nevýhody RBF:

- Je obtížné vymezit optimální střed funkcí
- Výsledná síť má často slabou schopnost reprezentovat souhrnné vlastnosti dat [12]

Algoritmus RBFN

RBFN (Radial Basis Function Network) je síť tvořená metodou k-means. Tato metoda poskytuje „středy" pro skrytou vrstvu. Výstupní vrstva je poté trénována jako jednoduchý perceptron. V modelu je možné nastavit velikost skryté vrstvy změnou hodnoty *RBF clusters*.

Datový záznam (nebo vzorek) je reprezentován nejbližším shlukem (shluky). Jinými slovy, aktivační hodnota je nejvyšší pro nejbližší shluky. Když je vyhodnocena vzdálenost mezi záznamem a středem shluku, je tento rozdělen délkou, nebo šířícím parametrem (jménem sigma). Sigma je ekvivalent směrové odchylky v normálním rozdělení. V Clementine se hodnota sigma shluku rovná průměrné vzdálenosti tohoto shluku od dvou nejbližších. Hodnotu sigma lze efektivně zvýšit i snížit úpravou hodnoty *RBF overlapping*. Pokud je hodnota větší než 1, budou shlukem reprezentovány záznamy dále od středu shluku. Při snížení této hodnoty (směrem k nule), bude shluk reprezentovat pouze záznamy, které jsou těsně u něj. [12]

4.6.3 Prevence „přetrénování" (Over-training)

V uzlu *Neural net* existuje volba *Prevent overtraining*. Tato volba umožní rozdělení dat na trénovací a testovací podmnožinu. Velikost těchto podmnožin určuje hodnota *Sample* v procentech. Trénovací data jsou použita pro stavbu modelu a testovací data jsou použita pro měření chyb modelu na různých skrytých datech. Tento cyklus je mnohokrát opakován a model s nejmenší chybovostí je vybrán jako nejlepší. Na konci trénování modelu se tato nejlepší síť zobrazí v záložce *Model manager*. [12]

4.6.4 Chybějící hodnoty v neuronových sítích

Uzel *Neural Net* potřebuje platné hodnoty na všech vstupních polích. Pokud některý ze vstupů obsahuje chybějící hodnoty, tak jsou převedeny na hodnoty podle níže uvedené tabulky. Uzel provádí tyto konverze automaticky. Proto je důležité rozhodnout, který záznam a pole by se k tvorbě modelu neměly použít. Kromě toho uživatel Clementine může kontrolovat konverze pomocí data preparation uzlů než ve streamu použije uzel *Neural net*(viz příklad monitoringu strojové soustavy). V následující tabulce je zobrazen přehled typů polí a k nim příslušné konverze. [12]

Tabulka 9. Konverze chybějících hodnot neuronové sítě

Typ pole	**Nepovolená/chybějící hodnota**	**Nepovolená hodnota převedena na:**
Flag	Chybějící nebo nějaká neznámá hodnota	0.5 (střední hodnota 0 a 1). Pole Flag je v NS reprezentováno hodnotou 0 a 1
Sets	Chybějící nebo nějaká neznámá hodnota	0 (pro všechny hodnoty 0,1 vstupních polí, které NS vytváří k reprezentaci vstupního pole typu Set)
Range	Chybějící hodnota nebo hodnota větší než horní mez	Nahrazeno horní mezí
Range	Méně než dolní mez	Nastavena dolní mez
Range	Chybějící hodnota nebo nenumická hodnota	Střední hodnota rozmezí

5. Fáze přípravy dat

Tato fáze je nejobtížnější a časově nejnáročnější z celého procesu data miningu. Má klíčový význam pro úspěch dané úlohy. Dala by se popsat jako účelová úprava dat podle definovaných cílů. V Clementine se zdrojová data upravují pomocí uzlů, které nalezneme v záložkách *Record Ops* (operace se záznamy) a *Field Ops* (operace s poli). Cíle data DP jsou zpravidla dvojí: [12]

- Je potřeba vybrat (nebo vytvořit) z dostupných dat takové údaje, které jsou relevantní pro danou úlohu
- Upravit tyto údaje v podobě, která je vhodná pro zpracování vhodným algoritmem.

Postup této fáze vypadá následovně: [12]

- Popis datové množiny
- Výběr a vyčištění dat
- Konstrukce a sloučení dat
- Formátování dat

Na konci fáze **DP** získáme datovou matici. **Řádky** této matice obsahují sledované objekty, **sloupce** pak obsahují vhodné vlastnosti těchto objektů (vstupy, výstupy, identifikátory). [12]

5.1 Popis datové množiny

Kromě dat uchovávaných v relačních tabulkách, u kterých nezáleží na pořadí objektů (objekty jsou navzájem nezávislé), můžeme v řadě oblastí narazit na data s určitou strukturou. K takovým datům patří:

- Časová data (např. časové řady kurzů akcií)
- Prostorová data (např. geografické informační systémy)
- Strukturální data (např. chemické sloučeniny)

Časová data jsou obvykle tvořena hodnotami téže veličiny (nebo více veličin) zaznamenávaných v různých okamžicích. Veličinou (typicky numerickou) může být např.

kurz ceny akcií, spotřeba elektrické energie, výsledek laboratorního testu v nemocnici, transakce na účtu v bance, nebo sekvence navštívených webových stránek. Údaje tedy mohou být zaznamenávány jak v pravidelných časových intervalech (denní kurzy akcií), nebo v libovolných okamžicích (návštěvy u lékaře). Obvyklou úlohou pro časová data je predikce budoucí hodnoty veličiny (např. kurzu akcií) na základě hodnot zjištěných v minulosti. Z časové řady se postupně vybírá úsek dané délky tak, že jeho „začátek" představuje hodnoty vstupních atributů a jeho „konec" hodnoty cílového atributu.

Jiným příkladem časově závislých dat a jejich využití je situace, kdy tato data představují jen dílčí informace popisující složitější objekt. Příkladem může být úloha diagnostikování choroby, mimo jiné na základě opakovaných vyšetření. Pak se sekvence výsledků téhož laboratorního testu chápe jako relace *n:1* ve vztahu k pacientovi.

Prostorová data se využívají v případech, kdy mezi jednotlivými objekty hraje roli implicitní relace sousednosti. Obvykle předpokládáme, že hodnoty atributů u sousedících objektů nebudou příliš odlišné. Většina analytických metod nedokáže tuto vazbu mezi objekty vzít do úvahy. Spíše se tedy použije při interpretaci, nebo jako dodatečné požadavky na nalezené znalosti.

Strukturální data se v DP převádějí na hodnoty atributů. Jedná se o konstrukce nových atributů a konstrukce jedné relace (tabulky) z více vzájemně propojených tabulek (4. krok fáze DP).

5.2 Výběr dat a jejich vyčištění

Ne všechny atributy jsou relevantní pro zamýšlenou analýzu. V reálných úlohách se setkáváme s databázemi, které obsahují desítky až stovky atributů. Je potřeba tyto atributy nějakým způsobem zredukovat. Nabízejí se nám dva způsoby řešení:

- **Transformace** – Z existujících atributů se vytvoří menší počet atributů nových
- **Selekce** – Výběr jen těch „nejdůležitějších" atributů vhodných pro danou úlohu

Pro transformaci atributů se používá např. Karhounen Loevův rozvoj, faktorová analýza nebo analýza hlavních komponent. Tyto metody předpokládají použití výhradně numerických atributů. Základem všech uvedených metod je reprezentace objektů pomocí hodnot nových atributů, které vzniknou jako lineární kombinace atributů původních.

Nevýhodou je, že potřebujeme znát (měřit) hodnoty všech původních atributů. Navíc nové transformované atributy nemusí mít srozumitelnou interpretaci.

Při (automatizované) selekci atributů obvykle hledáme ty atributy, které nejlépe přispějí ke klasifikaci objektů do tříd. Máme dvě možnosti, jak k dané úloze přistoupit. Můžeme použít tzv. *metodu filtru*(filter approach) neboli spočítat pro všechny atributy nějakou charakteristiku, vyjadřující vhodnost pro klasifikaci, nebo použít nějaký algoritmus strojového učení pro vytvoření modelu z (pod)množiny atributů a vyhodnotit model (tzv. *metoda obálky* wrapper approach).

Po transformaci a selekci atributů je na řadě čištění dat. V této fázi se vhodným způsobem zbavíme polí a proměnných, které nechceme při tvorbě modelu použít. V Clementine nám k tomuto účelu slouží uzel *Filter*, kde nastavíme, které položky nechceme v datovém proudu dále využít.

5.3 Konstrukce a sloučení dat

Existují dvě možnosti konstrukce dat. Agregace proměnných z více vzájemně propojených tabulek nebo odvození atributů z původních dat. Operace, která umožní z jedné nebo více relací (tabulek) vytvořit relaci novou, se v relační algebře nazývá *spojení* (join). Tato operace musí brát do úvahy různý typ vztahů mezi entitami ve spojovaných tabulkách.

Základními typy vztahů z hlediska kardinality jsou vztah 1:1, vztah 1:n (n:1) a vztah n:m. V případě spojení 1:1 vznikne nová relace, která bude obsahovat sloupce z obou relací a počet řádků bude odpovídat počtu řádků v první relaci.
V případě spojení 1:n se vytvoří tabulka, která bude obsahovat nové agregované hodnoty získané z *n* opakování údajů, vztahujících se k entitě na straně *„1“* vztahu mezi relacemi. Těmito agregovanými hodnotami (novými atributy) bývají (podle typu původních atributů):

- **Pro numerické atributy**: Součet, minimum, maximum, průměr, …
- **Pro kategoriální atributy**: Počet různých hodnot, výskyt konkrétní hodnoty, majoritní hodnota, …

Počet řádků v nové tabulce pak bude odpovídat počtu řádků v relaci na straně „1“. Vztah n:m se obvykle vyjadřuje pomocí další relace, která je s původními relacemi svázána vztahem 1:n resp 1:m. Převedeme tedy vztah n:m podle úlohy na vztah 1:m nebo n:1. Podle

toho, jakou relaci chceme zkoumat, zvolíme ji jako primární, a spojení se bude provádět vzhledem k této vybrané relaci. Typ agregovaných hodnot i počet řádků ve výsledné tabulce bude záviset na této volbě.

Z atributů přítomných v původních datech lze samozřejmě také vytvářet odvozené atributy. Někdy tvorbu nových atributů vyžaduje způsob předzpracování. V příkladu monitoringu zkušebního provozu je tento způsob ukázán. Jindy tvorba atributů plyne z doménových znalostí (např. převod rodného čísla klienta na věk a pohlaví).

5.4 Formátování dat

Jako poslední úkon, který je potřeba ve fázi DP udělat, je formátování dat. Zde bereme ohled na výběr modelů, které chceme použít. Záleží na tom, zda modely umí pracovat se symbolickými výstupy, nebo ke své konstrukci využívají pouze numerických hodnot. Jisté omezení je i to, že některé modely využívají pouze kategoriální data. Numerické hodnoty tedy nejdřív musíme rozdělit na intervaly (diskretizovat). Některé modely provádí diskretizaci v průběhu práce algoritmu.

Závěr

Práce se zabývá problematikou, která se na Fakultě mechatroniky, informatiky a mezioborových studií dosud nepěstovala. Po akreditaci nových studijních informatických oborů vzniká ve studijních programech větší prostor pro nové informatické disciplíny. Mezi tyto disciplíny nesporně patří i problematika dataminingu. Tato diplomová práce je příspěvkem k přípravě studijních materiálů pro nový předmět, který by navazoval na předměty o databázích a rozšiřoval by tak spektrum informačních dovedností, které by absolventi mohli získat.

Prvním cílem bylo popsat koncept dataminingu. Tento cíl byl splněn v první kapitole, která obsahuje definici pojmu datamining a krátkou zmínku o historii tohoto odvětví. Dále jsou popsány úlohy datamimingu, jeho metodiky a typy zdrojů, se kterými pracuje. Práce nemá ambice úplného výčtu všech dnes známých a používaných dataminingových postupů, nýbrž má poskytnout základní popis a přehled. Detailněji se věnuje některým vybraným principům a postupům v dalších kapitolách.

Druhým cílem bylo vytvoření studijních materiálů k vybraným analytickým metodám a jejich konkretizace k modelům systému Clementine. V druhé kapitole jsou představeny modely a algoritmy rozhodovacích stromů. Postupně jsou popsány modely C5.0, CHAID, C&R Tree a QUEST.

Popis asociačních pravidel je uveden ve třetí kapitole. Na začátku je popsán základní tvar asociačního pravidla a jeho zobecněný tvar. Následuje popis a použití modelů Apriori, GRI a Carma. Nakonec je popsán systém generování kombinací a speciální případy asociačních pravidel.

Čtvrtá kapitola obsahuje obecné informace o neuronových sítích. Popis biologického neuronu a k němu je analogicky popsaný neuron umělý. Dále jsou popsány funkce jednotlivých algoritmů jediného uzlu neuronové sítě, dostupného v Clementine.

V poslední páté kapitole je krátce přiblížen proces přípravy dat k modelování. Tato kapitola je úzce spjata s příkladem neuronové sítě, obsaženým v přílohách diplomové práce.

Přílohy obsahují příklady k jednotlivým modelům vytvořené v systému Clementine, popsané jsou jednotlivé datové proudy a funkce použitých uzlů. Tam, kde to lze, (např. kvůli

velikosti výsledného modelu) je vytištěn výsledný vygenerovaný model. První příklad obsahuje všechny modely rozhodovacích stromů na zadání, které zjišťuje rizikového klienta pro bankovní úvěr. Vytvoření modelů a diskuse výsledků je pouze ilustrativní pro dataminingové postupy, protože data jsou pochopitelně smyšlená jen jako „školní“ příklad. Podobně je to v příkladu s využitím asociačních pravidel. Zadání opět kopíruje dnes často diskutované marketingové téma, kterým je analýza nákupního košíku. Poslední příklad pracuje s modelem neuronové sítě pro další frekventovanou dataminingovou úlohu monitorování zkušebního provozu nového stroje. Opět je třeba zdůraznit, že data jsou smyšlená a upravena pro zadání. Konkrétní provozní data bych v daném čase nemohl získat.

Uvedené příklady jsou sice učebnicové, ale k účelu vysvětlení problematiky práce v Clementine postačují.

Představené modely a postupy však mohou být v dalších projektech nebo závěrečných prácích aplikované na reálná data, pokud se je podaří získat. Na přiloženém disku DVD je nahrán vytvořený manuál pro základy práce v Clementine.

Použitá literatura

[1] BERKA, Petr. *Dobývání znalostí z databází.* 1. vyd. Praha: Academia, 2003. 366 s. ISBN 80-200-1062-9

[2] LACKO, Luboslav. *Databáze: datové sklady, OLAP a dolování dat s příklady v Microsoft SQL Serveru a Oracle.* 1. vyd. Brno: Computer Press, 2003. 486 s. ISBN 80-7226-969-0.

[3] LACKO, Luboslav. *Business Intelligence v SQL Serveru 2005 Reportovací, analytické a další datové služby.* 1. vyd. Brno: Computer press, 2006. 391 s. ISBN 80-251-1110-5

[4] Olivia Parr Rud. Datamining, praktický průvodce dolováním dat pro efektivní prodej, cílený marketing a podporu zákazníků (CRM). 1. vyd. Praha: Computer press, 2001. 329 s. ISBN 80-7226-577-6

[5] CRM portal, CRM a Business Intelligence: Redakce [online]. Dostupné z WWW: <http://www.crmportal.cz/redakcni/crm-a-business-intelligence>

[6] Dobývání znalostí z databází pro účely business intelligence [online]. Dostupné z WWW: < http://www.lukassykora.cz/content/dob%C3%BDv%C3%A1n%C3%AD-znalost%C3%AD-z-datab%C3%A1z%C3%AD-pro-%C3%BA%C4%8Dely-business-intelligence>

[7] Martin Plchút, Dobývání znalostí z databází [online]. Dostupné z WWW: < http://www.fit.vutbr.cz/study/courses/ZZD/public/seminar0304/Uvod.pdf>

[8] Roman Danel, Dobývání znalostí [online]. Datum publikování 2URL: < http://homel.vsb.cz/~dan11/isys/Danel%20-%20IS%20-%20Dolovani%20dat.pdf>

[9] Han J., Kamber M.: Datamining: Concepts and Techniques. Morgan Kaufmann, 2001, ISBN 1-55860-489-8.

[10] Petr Berka, Rozhodovací stromy [online]. 27.12.2010. Dostupné z WWW: < http://sorry.vse.cz/~berka/docs/izi456/kap_5.1.pdf>

[11] Petr Berka, Asociační pravidla [online]. Dostupné z WWW: < http://sorry.vse.cz/~berka/docs/izi456/kap_5.2.pdf >

[12] Tutorialy a firemní dokumentace PASW Modeler, dodavatel Predictive Analytics Software (PASW)

[13] Kódování bez šumu (zdrojové kódování), s šumem a sdělovací kanál [online]. Dostupné z WWW: < http://stag.zcu.cz/fel/kae/pi/kody/kodovani%20s%20sumem.pdf>

[14] Richard Hindls, Stanislava Hronová, Jan Seger. Statistika pro ekonomy, 5. vyd. Praha: Professional publishing, 2007. 415 s. ISBN 80-86419-59-2

[**15**] Petr Berka, Statistika [online]. Dostupné z WWW:
< http://sorry.vse.cz/~berka/docs/izi456/kap_3.pdf >
[**16**] Petr Berka, Neuronové sítě [online]. Dostupné z WWW:
< http://sorry.vse.cz/~berka/docs/izi456/kap_5.4.pdf >
[**17**] SEER's Training Website, Nervous System [online]. Dostupné z WWW:
< http://www.web-books.com/eLibrary/Medicine/Physiology/Nervous/Nervous.htm >
[**18**] Ivo Bukovský, Neuronové sítě (Úvod a MLP sítě) [online]. Dostupné z WWW:
<www.fsid.cvut.cz/cz/u12110/bukovsky/UI_1_prednaska_2005.ppt>

Přílohy

Příloha A – příklady

A.1 Příklad k rozhodovacím stromům

Typickým příkladem použití rozhodovacích stromů je případ, kdy banka (nebo jiný peněžní ústav) zjišťuje, kterým současným klientům nabídnout nový produkt.

A.1.1 Úloha

Zjistit podle informací obsažených v databázi klientů, zda je zákazník vhodný pro nabídku nového produktu. Zjišťujeme rizikovost jednotlivých klientů.

A.1.2 Data

Zdrojový soubor tohoto cvičení je Risk.txt. Je to předem upravený soubor z pohledu fáze přípravy dat, který je využíván jako cviční příklad v Clementine. Proto v datovém proudu nejsou žádné uzly, které upravují data pro tvorbu modelu (více viz kapitola Fáze přípravy dat). Jednotlivá pole zdrojového souboru a jejich popis je uveden v následující tabulce.

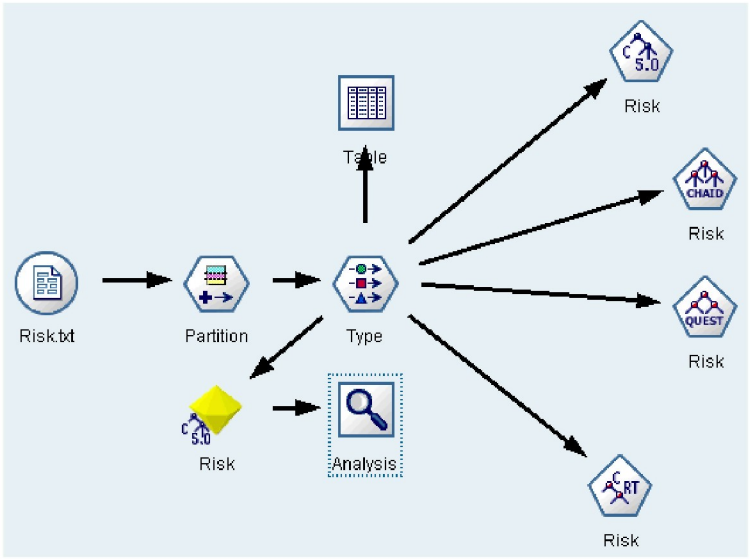
Název	**Popis**
ID	ID klienta (umělý klíč)
Vek	Věk klienta
Prijem	Příjem klienta
Pohlavi	Pohlaví (m,f)
Status	Rodinný stav (married, single)
PocDeti	Počet dětí
PocKaret	Počet karet
JakPlaceno	Splátky (monthly, weekly)
Hypoteka	Hypotéka (Ano,Ne)
PocAut	Počet aut ve vlastnictví klienta
Pujcky	Počet aktivních půjček klienta
Riziko	Riziko u klienta

A.1.3 Stream

Datový proud obsahuje následující uzly:

- *Var. file* „Risk.txt“ – výše popsaný zdrojový soubor

- Uzel *Partition* - rozděluje množinu dat na data trénovací a testovací, případně validační, podle paramatrů, které si uživatel nastaví v "Settings". V tomto případě jen na trénovací a testovací.
- Uzel *Type* – Rozděluje pole na vstupní a výstupní. V našem příkladu jsou všechna pole nastavená jako vstupní. Jediné výstupní pole je zde pole *Riziko*
- Uzel *Table* – zobrazuje vytvořenou datovou matici, ze které se vytvářejí modely.
- Následují 4 uzly nastavení parametrů pro tvorbu jednotlivých modelů rozhodovacích stromů. Postupně shora model C5.0, CHAID, QUEST, C&R Tree
- Uzel ve tvaru diamantu *Risk* je vytvořený model rozhodovacího stromu algoritmem C5.0.
- Na uzel modelu je napojen výstupní uzel *Analysis*, který analyzuje kvalitu predikčního modelu rozhodovacího stromu.



Obr. A.1.1 Stream příkladu Risk

Výsledek uzlu *Analysis* je zobrazen na následujícím obrázku. Je vidět, že trénovací data jsou správná pouze ze 78,32% a testovací dokonce jen ze 74,28%. Tato čísla nám ukazují skutečnost, že vytvořené modely by neměly být použity k rozhodování. Nesplňují totiž minimální hranici spolehlivosti modelu, která je 85%.

Analysis

Results for output field Risk

Comparing $C-Risk with Risk

'Partition'	1_Training		2_Testing	
Correct	1 568	78,32%	1 571	74,28%
Wrong	434	21,68%	544	25,72%
Total	2 002		2 115	

Obr. A.1.2 Výsledek uzlu Analysis

Výsledné modely jsou pro ukázku v textovém souboru příliš velké. Jsou uloženy na přiloženém cd k diplomové práci. Pro představu, jak vypadá model rozhodovacího stromu, vytvořený algoritmem C5.0 vás odkazuji na příklad k neuronové síti.

A.2 Příklad asociačních pravidel

Typickým příkladem pro asociační pravidla je analýza nákupního košíku. Jednu takovou si ukážeme.

A.2.1 Úloha

Máme za úkol zjistit, které produkty si zákazník pravděpodobně koupí v závislosti na tom, které produkty už zakoupil (resp. jsou v nákupním košíku) a jakou pravděpodobnost mají tyto kombinace asociačních pravidel.

A.2.2Data

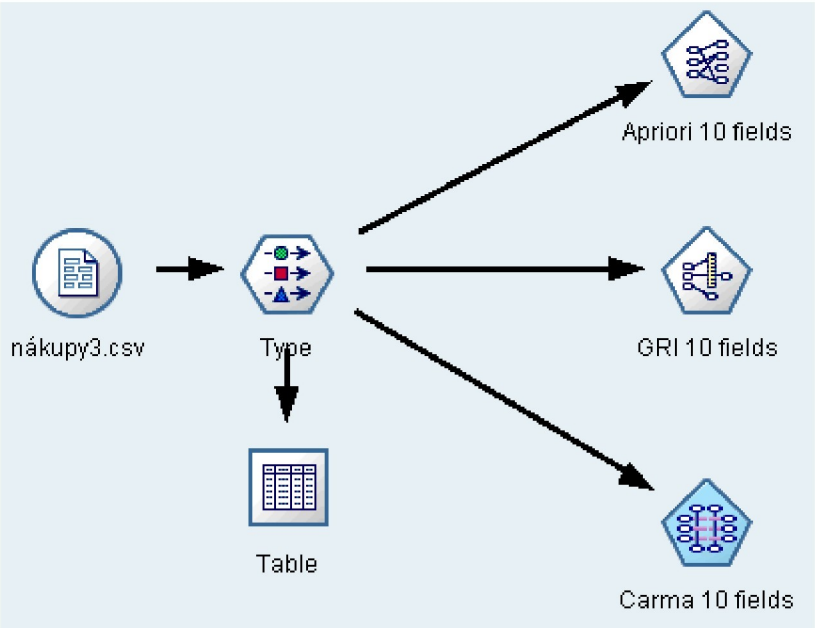
Zdrojový soubor „nákupy.txt“ obsahuje jednotlivé položky sortimentu obchodu a informace o zákaznících. Mezi položky sortimentu patří *Polotovary, Mražené jídlo, Alkohol, čerstvá zelenina* atd. Tato pole jsou označeny indikátorem 1 nebo 0 (typ pole *flag*) podle toho, zda si zákazník danou položkou koupil či nikoliv. Informace o klientech obsahují tato pole:

Pole	**Popis**
POHLAVI	Pohlaví
Vek	Věková skupina
Status	Rodinný stav
Deti	Zda má zákazník děti (Y/N)
Pracujici	Zda je zákazník zaměstnaný (Y/N)

A.2.3 Stream

I v tomto případě je datový proud velmi krátký ze stejného důvodu jako u příkladu s rozhodovacími stromy. Zdrojový soubor není potřeba nijak upravovat. Jednotlivé uzly streamu jsou tyto:

- *Var. file* „nákupy.txt“ – výše popsaný zdrojový soubor
- *Type* – uzel ve kterém nastavíme směr dat v datovém proudu. Tentokrát pro tvorbu modelů asociačních pravidel využíváme pouze pole sortimentu (tedy obsah košíku). Tato pole jsou nastavena jako vstupní i výstupní (ve sloupci Direction položka *Both*). Pole, která patří k informacím o zákazníkovi (viz tabulka výše) jsou pro tvorbu modelu nepotřebná, tedy ve sloupci Direction nastavíme *None*. Výsledná datová matice pro tvorbu modelů tato pole obsahovat nebude. Stejného výsledku bychom dosáhli vymazáním příslušných sloupců ve zdrojovém souboru.
- *Table* – zobrazuje datovou matici použitou k tvorbě modelů
- *Apriori 10 fields* – Uzel pro nastavení parametrů algoritmu Apriori
- *GRI 10 fields* – nastavení parametrů algoritmu GRI
- *Carma 10 fields* – nastavení parametrů algoritmu Carma



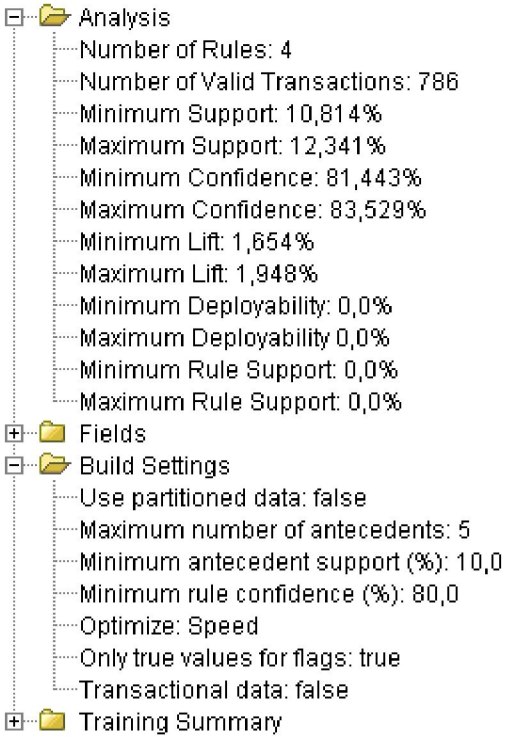
Obr. A.2.1 Stream analýzy nákupního košíku

A.2.4 Výsledky modelu apriori

Výsledkem vytvořeným algoritmem apriori (při základním nastavení) je soubor čtyř asociačních pravidel (viz obr.)

Consequent	Antecedent	Support %	Confidence %
Pecivo	Mleko Mrazene_jidlo	10,814	83,529
Pecivo	Alkohol Konzervy Polotovary	12,087	83,158
Pecivo	Mrazene_jidlo Konzervy Pochutiny	11,45	82,222
Polotovary	Alkohol Konzervy Pecivo	12,341	81,443

Obr. A.2.2 Výsledek modelu Apriori



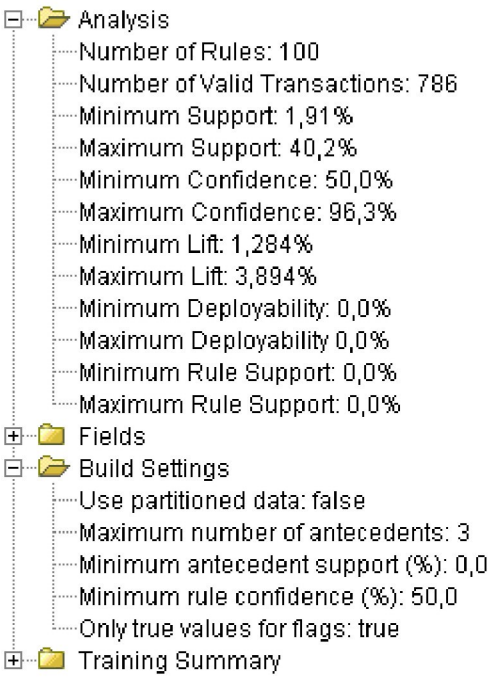
Obr. A.2.3 Shrnutí modelu Apriori

A.2.5 Výsledky modelu GRI

U tohoto modelu je výsledek obdobný s tím, že vytvořených pravidel je dohromady 100. Takže na dalším obrázku je jen vzorek těchto pravidel.

Consequent	Antecedent	Support %	Confidence %
Konzervy	Polotovary Cerstva_zelenina Pochutiny	3,44	96,3
Konzervy	Alkohol Cerstva_zelenina Pochutiny	3,05	95,83
Pecivo	Alkohol Cerstva_zelenina Mleko	2,54	95,0
Pecivo	Cerstva_zelenina Mleko Konzervy	3,56	92,86
Konzervy	Mrazene_jidlo Cerstva_zelenina Pochutiny	3,44	92,59

Obr. A.2.4 Výsledek modelu GRI



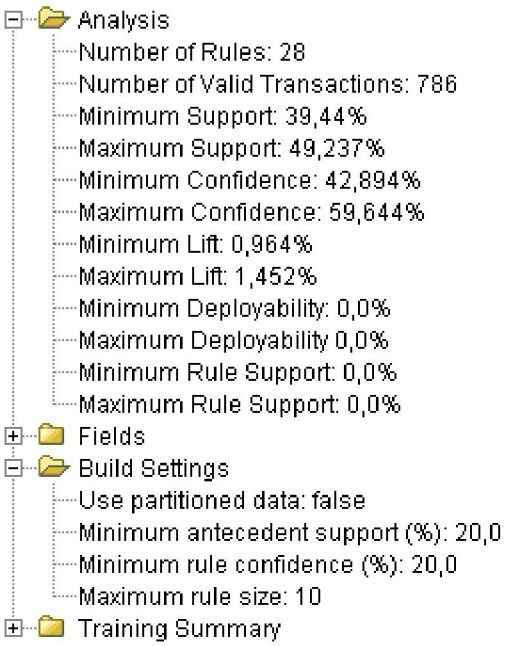
Obr. A.2.5 Shrnutí modelu GRI

A.2.6 Výsledky modelu Carma

Algoritmus Carma při základním nastavení vytvořil 28 asociačních pravidel.

Consequent	Antecedent	Support %	Confidence %
Polotovary	Pecivo	42,875	59,644
Mrazene_jidlo	Alkohol	39,44	58,387
Alkohol	Mrazene_jidlo	40,204	57,278
Pochutiny	Alkohol	39,44	55,484
Pecivo	Mrazene_jidlo	40,204	55,063
Pecivo	Alkohol	39,44	54,516
Pochutiny	Pecivo	42,875	54,303
Polotovary	Alkohol	39,44	53,871
Pochutiny	Mrazene_jidlo	40,204	53,165
Konzervy	Pecivo	42,875	53,116
Polotovary	Mrazene_jidlo	40,204	52,532
Pecivo	Polotovary	49,237	51,938
Mrazene_jidlo	Pecivo	42,875	51,632
Konzervy	Mrazene_jidlo	40,204	51,582
Polotovary	Pochutiny	47,455	51,475
Alkohol	Pecivo	42,875	50,148
Pecivo	Konzervy	45,547	50,0
Pochutiny	Polotovary	49,237	49,612
Pochutiny	Konzervy	45,547	49,162
Pecivo	Pochutiny	47,455	49,062
Polotovary	Konzervy	45,547	47,486
Konzervy	Pochutiny	47,455	47,185
Alkohol	Pochutiny	47,455	46,113
Mrazene_jidlo	Konzervy	45,547	45,531
Mrazene_jidlo	Pochutiny	47,455	45,04
Konzervy	Polotovary	49,237	43,928
Alkohol	Polotovary	49,237	43,152
Mrazene_jidlo	Polotovary	49,237	42,894

Obr. A.2.6 Výsledek modelu Carma



Obr. A.2.7 Shrnutí modelu Carma

A.3 Příklad na neuronové sítě

Příklad, na kterém si ukážeme model neuronové sítě, se týká monitoringu zkušebního provozu přístroje. Zavádění nových technologií do výroby je spojeno s určitými riziky. Především se testují na selhání.

A.3.1Úloha

Firma chce do výroby zavést nový přístroj a rozhodla se ho prozkoušet. Nějakou dobu zaznamenávala různé údaje o jeho provozu, chybová hlášení a jeho opravdový stav. Na základě těchto dat chce predikovat, za jakých podmínek dojde k poruše přístroje.

A.3.2 Data

Datový zdroj obsahuje několik časových řad záznamů o stavu stroje. Každý záznam je složen z těchto informací:

Název	Popis
Pořadové číslo	Čas, kdy byl proveden záznam
Výkon	Výkon stroje
Teplota	Teplota stroje
Tlak	0 pro normální stav, 1 pro výstražný stav tlaku
Čas	Čas od poslední kontroly
Status	0 normální stav, 101, 202, 303 chybové stavy
Výsledek	Stav do kterého systém dojde (0, 101, 202, 303)

A.3.3 Stream

- DERIVE *Varovani_tlaku* sčítá počet výstražných stavů tlaku pro jednotlivé časové řady zvlášť
- *Rozdil_teplot* počítá rozdíly teplot ve dvou následujících záznamech
- *Rozdil_vykonu* analogicky s předchozím počítá rozdíly výkonu
- *Kolisani_vykonu(dále jako kv)* vrací hodnotu TRUE, pokud změna výkonu v tomto a předchozím záznamu byla opačná[6]
- *Stav_vykonu* reaguje na předchozí stav implicitně vrací hodnotu Stabilni. Pokud dvě po sobě jdoucí hodnoty proměnné Kolisani_vykonu jsou TRUE, změní hodnotu na Kolisave. Do stavu Stabilni se vrací, pokud hodnoty proměnné Kolisani_vykonu jsou FALSE v posledních pěti záznamech.
- *Zmena_vykonu* počítá průměr z posledních pěti hodnot proměnné *Rozdil_vykonu*
- *Zmena_teploty* analogicky s proměnnou *Rozdil_teplot*

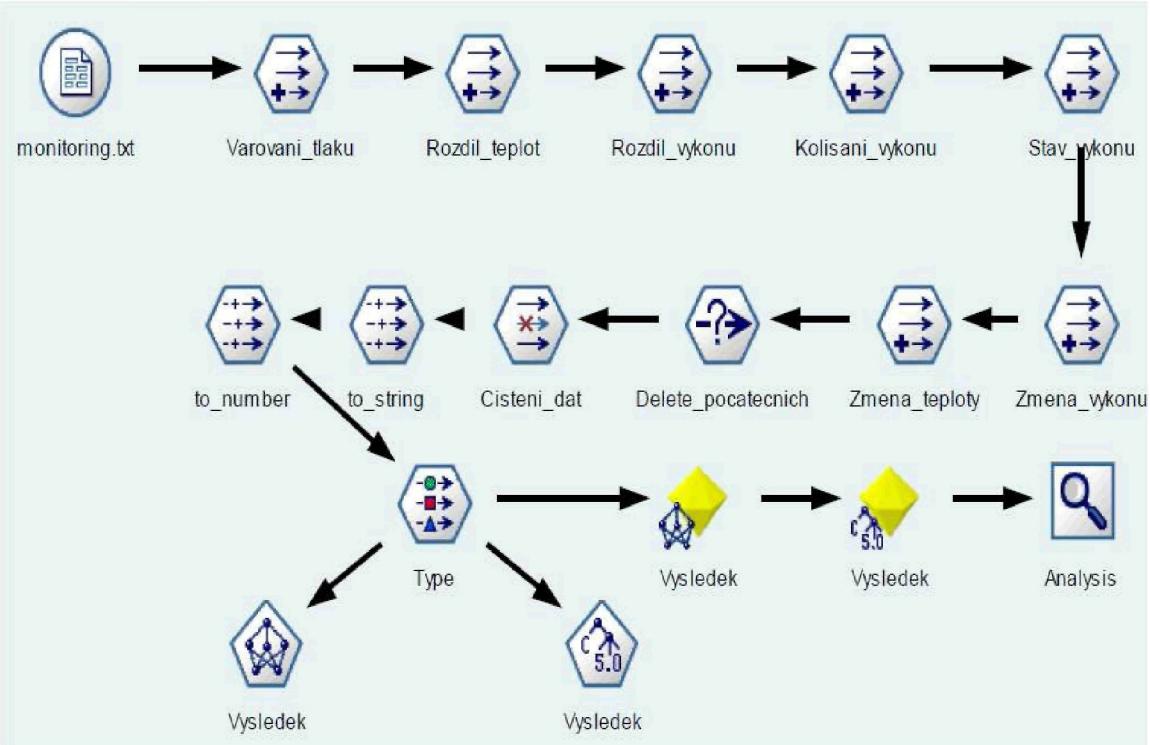
- Uzel SELECT s názvem *Delete_pocatecnich* smaže první záznam z každé časové řady, protože odvozené diferenční charakteristiky v nich nejsou definované

[6] Z kladné na zápornou nebo opačně

- Uzel FILTER *Cisteni_dat* odfiltruje všechny proměnné kromě Cas, Status, Vysledek, Varovani_tlaku, Stav_vykonu, Zmena_vykonu a Zmena_teploty

- Uzel FILLER *to_string* převádí číselné hodnoty proměnné *Vysledek* na řetězce
- *to_number* převádí hodnoty proměnné Varovani_tlaku na integer

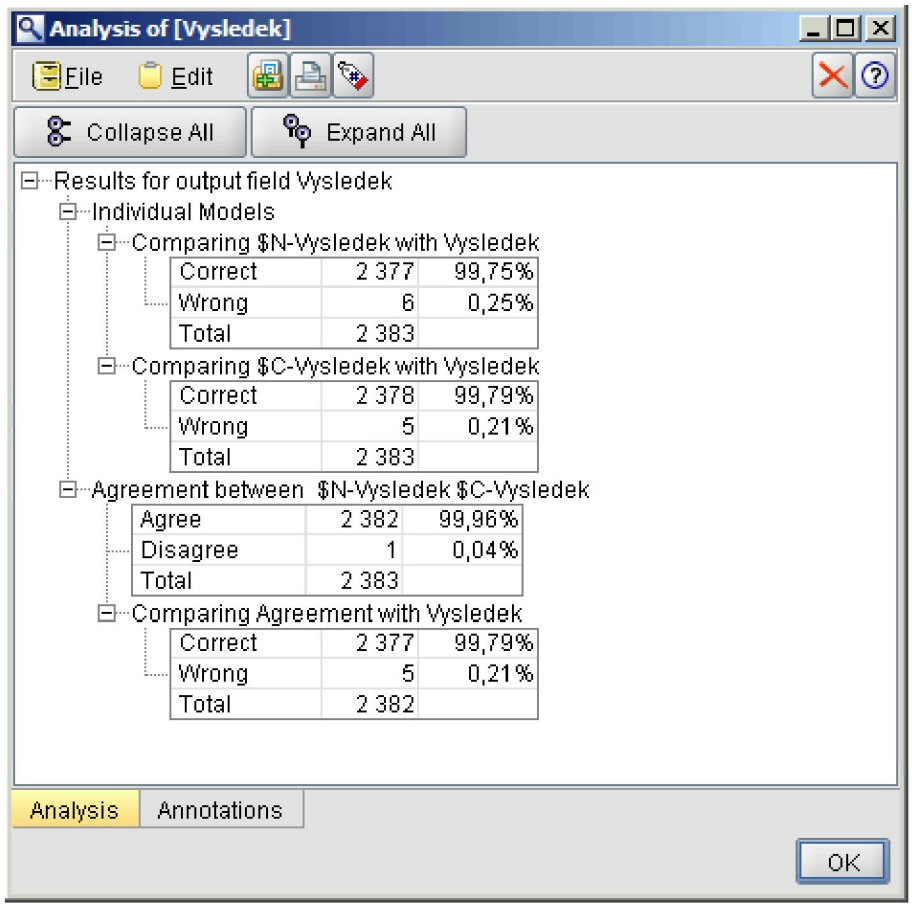
- V uzlu TYPE, nastavíme proměnnou *Vysledek* jako výstupní hodnotu (Direction OUT) a ostatní proměnné jako vstupní (Direction IN)
- Na uzel TYPE napojíme dva modely, C5.0 a Neural Net



Obr. A.3.1 Stream monitoringu stroje

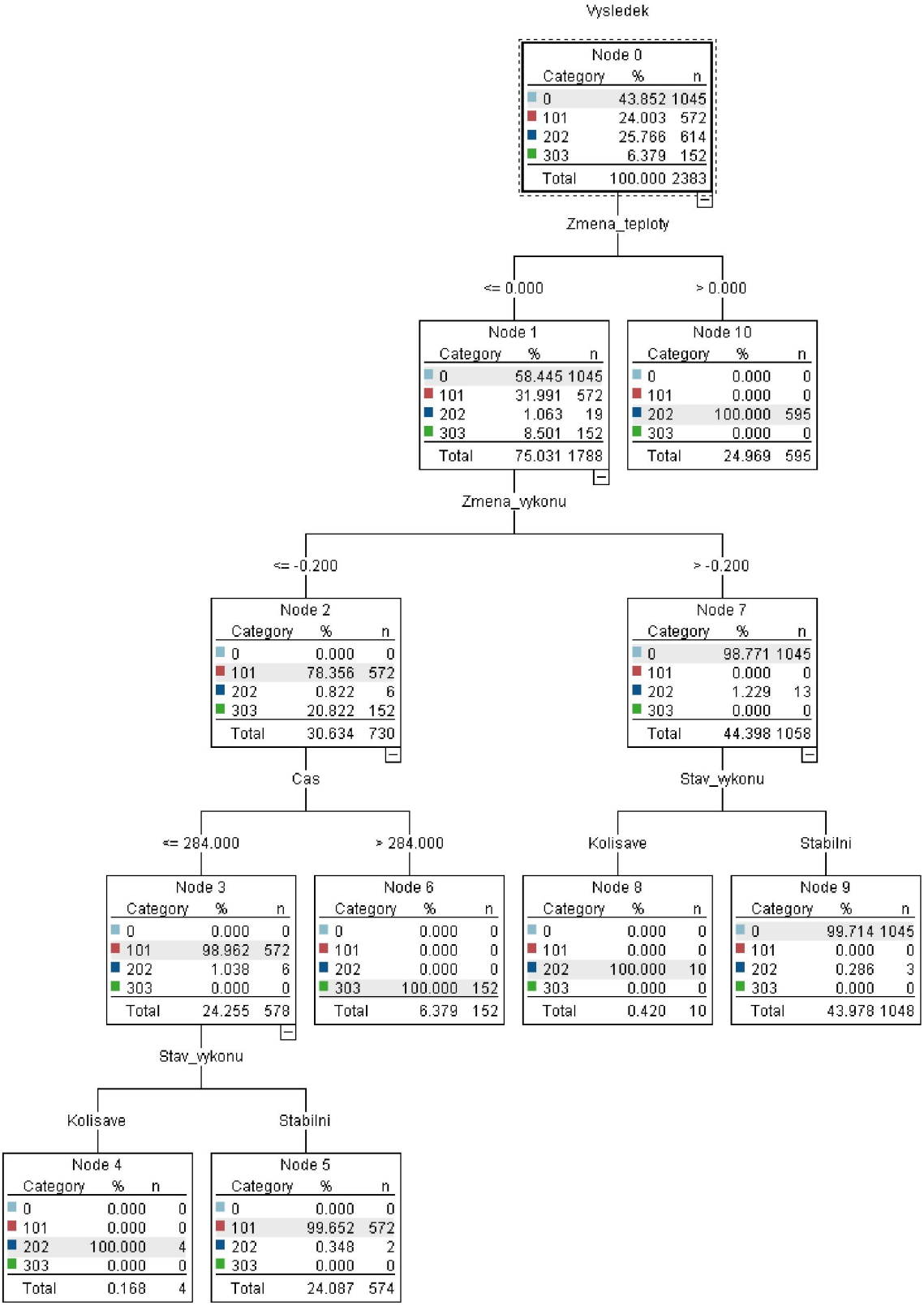
Po spuštění celého streamu se vytvoří dva modely. Rozhodovací strom C5.0 a Neuronová síť. Tyto modely (ikonky diamantu) vložíme do proudu a otestujeme si je pomocí výstupního uzlu Analysis. Výsledkem jsou dvě tabulky, ve kterých je srovnání výstupů jednotlivých modelů s modelovanou proměnnou *Vysledek* a porovnání výsledků rozhodovacího stromu a neuronové sítě. Tyto tabulky si můžeme prohlédnout na následujícím obrázku. V nich vidíme, že výstup rozhodovacího stromu se v 99,79%

shoduje s cílovou proměnnou Vysledek. Neuronová síť dosáhla prakticky stejného výsledku s 99,75%.



Obr. A.3.2 Analýza výstupních modelů

Výhodou rozhodovacího stromu je jeho jednoduchá interpretace. Prohlížením modelu na záložce Viewer je tento strom zobrazený (viz. obrázek). Závěr, který můžeme interpretovat na jeho základě je ten, že vzrůstající teplota je stoprocentní indikátor chybového stavu 202. Chybový stav 303 nastává, pokud teplota neroste (Zmena_teploty <= 0), výkon výrazně klesá (Zmena_vykonu <= -0,2), poslední kontrola byla provedena před 284 dny (Cas <= 284) a výkon je stabilní (Stav_vykonu = Stabilni).



Obr. A.3.3 Model rozhodovacího stromu C5.0

Příloha B – Manuál k programu Clementine

Druhou přílohou je manuál pro základy práce v systému Clementine. Obsahuje popis aplikace, jednotlivých typů uzlů a systém tvorby datového proudu. Vzhledem k velikosti byl umístěn na přiloženém CD.

Obsah CD

Přiložené cd k diplomové práci obsahuje následující soubory:

- Elektronickou verzi diplomové práce v souboru PDF
- Manuál práce s Clementine
- Zdrojové soubory k příkladům
- Soubory datových proudů (streamů) pro program Clementine

Přílohy – Příklady v SPSS PASW Modele

P.1 Riziko u klienta banky

V druhé kapitole diplomové práce jsou popsány rozhodovací stromy za použití příkladu z bankovnictví. V tomto příkladu zjišťujeme, zda je možné klientovi udělit úvěr. První příklad, na kterém si předvedeme modely rozhodovacích stromů je podobného ražení.

P.1.1 Úloha

Máme k dispozici data o klientech a naším úkolem je zjistit rizikovost klienta pro banku při udělení úvěru.

P.1.2 Data

Zdrojový soubor tohoto cvičení je Risk.txt. Obsahuje informace o klientech například, zda má klient hypotéku, případně jestli vlastní auto atd. Pole riziko je předem připravené výstupní pole z pohledu fáze přípravy dat. Naším úkolem bude také zjistit, do jaké míry jsou tato data spolehlivá. Jednotlivá pole zdrojového souboru a jejich popis je uveden v následující tabulce:

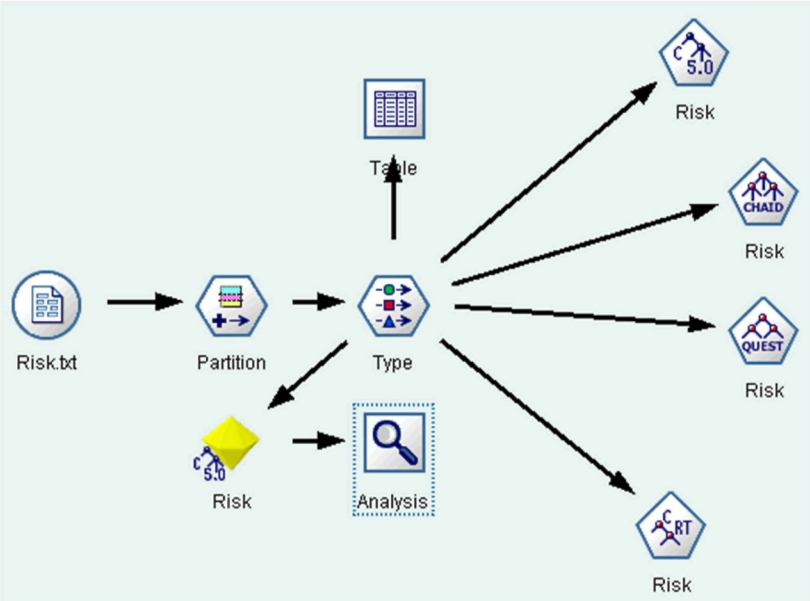
Název	**Popis**
ID	ID klienta (umělý klíč)
Vek	Věk klienta
Prijem	Příjem klienta
Pohlavi	Pohlaví (m,f)
Status	Rodinný stav (married, single)
PocDeti	Počet dětí
PocKaret	Počet karet
JakPlaceno	Splátky (monthly, weekly)
Hypoteka	Hypotéka (Ano,Ne)
PocAut	Počet aut ve vlastnictví klienta
Pujcky	Počet aktivních půjček klienta
Riziko	Riziko u klienta

P.1.3 Stream

Datový proud obsahuje následující uzly:

- *Var. file* „Risk.txt“ – výše popsaný zdrojový soubor
- Uzel *Partition* - rozděluje množinu dat na data trénovací a testovací, případně validační, podle paramatrů, které si uživatel nastaví v "Settings". V tomto případě jen na trénovací a testovací.

- Uzel *Type* – Rozděluje pole na vstupní a výstupní. V našem příkladu jsou všechna pole nastavená jako vstupní. Jediné výstupní pole je zde pole *Riziko*
- Uzel *Table* – zobrazuje vytvořenou datovou matici, ze které se vytvářejí modely.
- Následují 4 uzly nastavení parametrů pro tvorbu jednotlivých modelů rozhodovacích stromů. Postupně shora model C5.0, CHAID, QUEST, C&R Tree
- Uzel ve tvaru diamantu *Risk* je vytvořený model rozhodovacího stromu algoritmem C5.0.
- Na uzel modelu je napojen výstupní uzel *Analysis*, který analyzuje kvalitu predikčního modelu rozhodovacího stromu.



Obr. P.1.1 Stream příkladu Riziko klienta

Výsledek uzlu *Analysis* je zobrazen na následujícím obrázku. Je vidět, že trénovací data jsou správná pouze ze 78,32% a testovací dokonce jen ze 74,28%. Tato čísla nám ukazují skutečnost, že zdrojový soubor není příliš spolehlivý. Minimální hranice spolehlivosti dat je 85%. Řešením by byla tvorba vlastního výstupního pole „Riziko“ podobně jak je tomu u příkladu udělení stipendia (viz. třetí příklad).

Analysis

Results for output field Risk

Comparing $C-Risk with Risk

'Partition'	1_Training		2_Testing	
Correct	1 568	78,32%	1 571	74,28%
Wrong	434	21,68%	544	25,72%
Total	2 002		2 115	

Obr. P.1.2 Analýza modelu C5.0

Výsledné modely jsou pro ukázku v textovém souboru příliš velké. Jsou uloženy na přiloženém cd k diplomové práci. Pro představu, jak vypadá model rozhodovacího stromu, vytvořený algoritmem C5.0 vás odkazuji na příklad k neuronové síti.

P.2 Analýza nákupního košíku

Metody asociačních pravidel zpopularizovaly s analýzou nákupního košíku. Proto v tomto příkladu jednu takovou analýzu vytvoříme.

P.2.1 Úloha

Máme za úkol zjistit, které produkty si zákazník pravděpodobně koupí v závislosti na tom, které produkty už zakoupil (resp. jsou v nákupním košíku) a jakou pravděpodobnost mají tyto kombinace asociačních pravidel.

P.2.2 Data

Zdrojový soubor „nákupy.txt“ obsahuje jednotlivé položky sortimentu obchodu a informace o zákaznících. Mezi položky sortimentu patří *Polotovary, Mražené jídlo, Alkohol, čerstvá zelenina* atd. Tato pole jsou označeny indikátorem 1 nebo 0 (typ pole *flag*) podle toho, zda si zákazník danou položkou koupil či nikoliv. Informace o klientech obsahují tato pole:

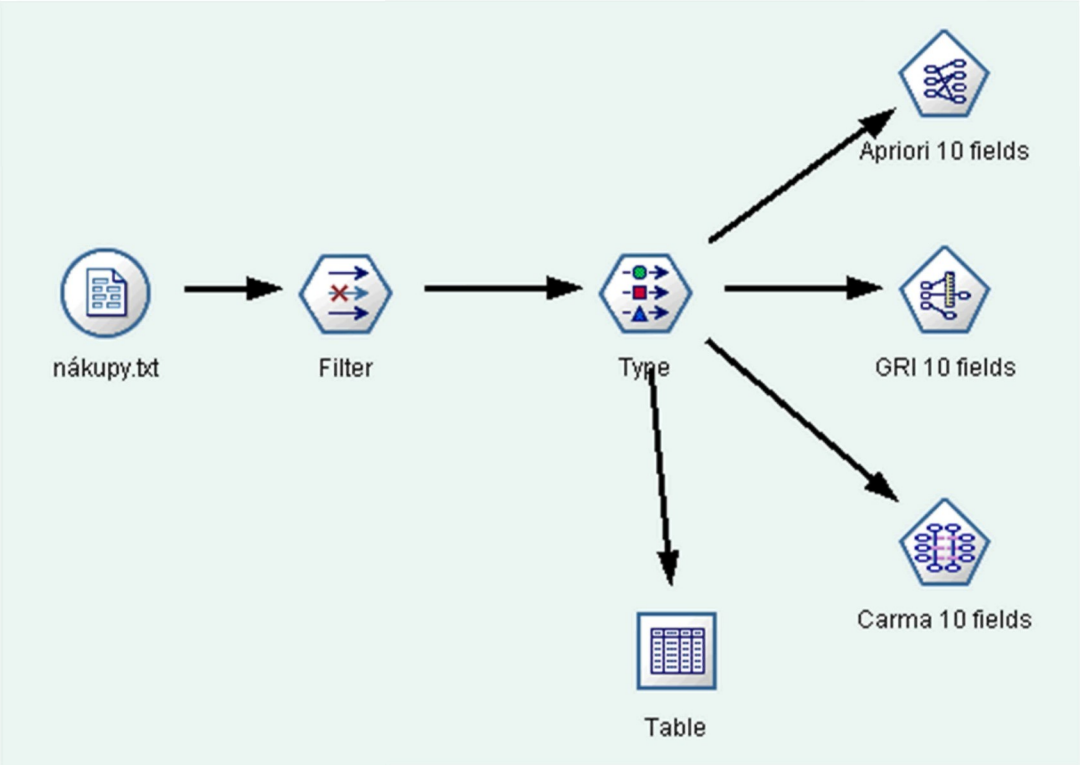
Pole	**Popis**
POHLAVI	Pohlaví
Vek	Věková skupina
Status	Rodinný stav
Deti	Zda má zákazník děti (Y/N)
Pracujici	Zda je zákazník zaměstnaný (Y/N)

P.2.3 Stream

I v tomto případě je datový proud velmi krátký ze stejného důvodu jako u příkladu s rozhodovacími stromy. Zdrojový soubor není potřeba nijak upravovat. Jednotlivé uzly streamu jsou tyto:

- VAR.FILE „nákupy.txt“ – výše popsaný zdrojový soubor
- Pomocí uzlu FILTER omezíme datovou matici pouze na relevantní pole využitá pro generování jednotlivých modelů (ponecháme pouze pole sortimentu)
- TYPE uzel ve kterém nastavíme směr dat v datovém proudu. Tentokrát pro tvorbu modelů asociačních pravidel využíváme pouze pole sortimentu (tedy obsah košíku). Tato pole jsou nastavena jako vstupní i výstupní (ve sloupci Direction položka *Both*). Pole, která patří k informacím o zákazníkovi (viz tabulka výše) jsou pro tvorbu modelu nepotřebná, tedy ve sloupci Direction nastavíme *None*. Výsledná datová matice pro tvorbu modelů tato pole obsahovat nebude. Stejného výsledku bychom dosáhli vymazáním příslušných sloupců ve zdrojovém souboru.

- *Table* – zobrazuje datovou matici použitou k tvorbě modelů
- *Apriori 10 fields* – Uzel pro nastavení parametrů algoritmu Apriori
- *GRI 10 fields* – nastavení parametrů algoritmu GRI
- *Carma 10 fields* – nastavení parametrů algoritmu Carma



Obr. P.2.1 Stream analýzy nákupního košíku

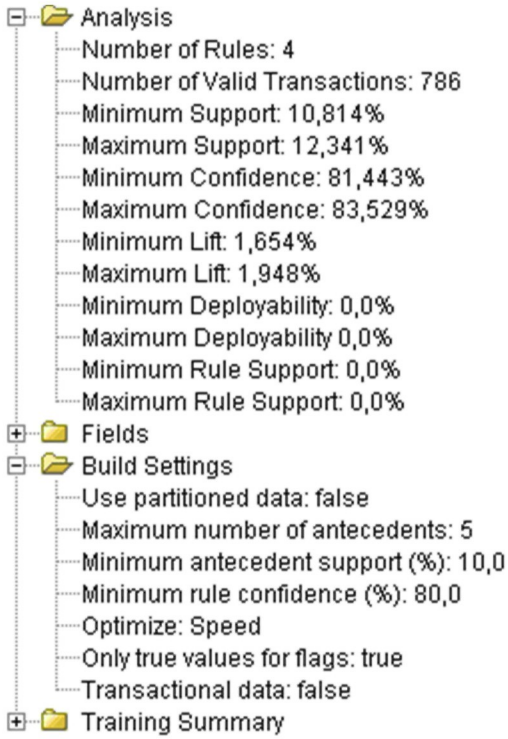
P.2.5 Výsledky modelu apriori

Výsledkem vytvořeným algoritmem apriori (při základním nastavení) je soubor čtyř asociačních pravidel (viz obr.)

Consequent	Antecedent	Support %	Confidence %
Pecivo	Mleko Mrazene_jidlo	10,814	83,529
Pecivo	Alkohol Konzervy Polotovary	12,087	83,158
Pecivo	Mrazene_jidlo Konzervy Pochutiny	11,45	82,222
Polotovary	Alkohol Konzervy Pecivo	12,341	81,443

Obr. P.2.2 Model Apriori

Největší podporu z těchto pravidel má poslední vypsané Alkohol&Konzervy&Pecivo -> Polotovary 12,341%. To znamená, že zmíněná trojice předpokladů se v datech vyskytuje ve 12 procentech případů a existuje 81,443% pravděpodobnost, že k nim zákazník zakoupí i polotovary. Přehled modelu apriori vypadá následovně:



P.2.6 Výsledky modelu GRI

Model GRI nám vrátil obdobné, ovšem poněkud zdrobnělé výsledky oproti modelu Apriori. Je to dáno především nastavením uzlu GRI při generování modelu. U tohoto modelu nás zajímají pravidla s největší spolehlivostí. V tomto modelu je to pravidlo Polotovary&Cerstva_zelenina&Pochutiny -> Konzervy se spolehlivostí 96,3%.

Consequent	Antecedent	Support %	Confidence %
Konzervy	Polotovary Cerstva_zelenina Pochutiny	3,44	96,3
Konzervy	Alkohol Cerstva_zelenina Pochutiny	3,05	95,83
Pecivo	Alkohol Cerstva_zelenina Mleko	2,54	95,0
Pecivo	Cerstva_zelenina Mleko Konzervy	3,56	92,86
Konzervy	Mrazene_jidlo Cerstva_zelenina Pochutiny	3,44	92,59

Obr. P.2.3 Model GRI

Přehled modelu GRI vypadá následovně:

- Analysis
  - Number of Rules: 100
  - Number of Valid Transactions: 786
  - Minimum Support: 1,91%
  - Maximum Support: 40,2%
  - Minimum Confidence: 50,0%
  - Maximum Confidence: 96,3%
  - Minimum Lift: 1,284%
  - Maximum Lift: 3,894%
  - Minimum Deployability: 0,0%
  - Maximum Deployability 0,0%
  - Minimum Rule Support: 0,0%
  - Maximum Rule Support: 0,0%
- Fields
- Build Settings
  - Use partitioned data: false
  - Maximum number of antecedents: 3
  - Minimum antecedent support (%): 0,0
  - Minimum rule confidence (%): 50,0
  - Only true values for flags: true
- Training Summary

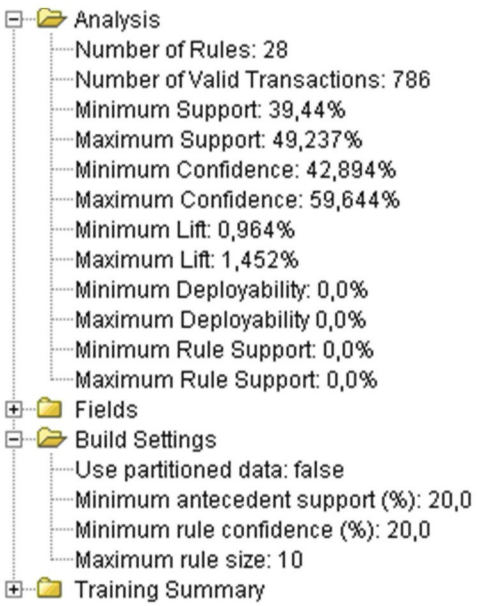
P.2.7 Výsledky modelu Carma

Algoritmus Carma při základním nastavení vytvořil 28 asociačních pravidel. Je vidět, že tento model je zaměřený především na jednoduchá pravidla ve tvaru Předpoklad-> Závěr. Všechny hodnoty v tomto modelu jsou velmi podobné, tak zmiňuji akorát pravidlo s největší spolehlivostí Pecivo -> Polotovary téměř 60%.

Consequent	Antecedent	Support %	Confidence %
Polotovary	Pecivo	42,875	59,644
Mrazene_jidlo	Alkohol	39,44	58,387
Alkohol	Mrazene_jidlo	40,204	57,278
Pochutiny	Alkohol	39,44	55,484
Pecivo	Mrazene_jidlo	40,204	55,063
Pecivo	Alkohol	39,44	54,516
Pochutiny	Pecivo	42,875	54,303
Polotovary	Alkohol	39,44	53,871
Pochutiny	Mrazene_jidlo	40,204	53,165
Konzervy	Pecivo	42,875	53,116
Polotovary	Mrazene_jidlo	40,204	52,532
Pecivo	Polotovary	49,237	51,938
Mrazene_jidlo	Pecivo	42,875	51,632
Konzervy	Mrazene_jidlo	40,204	51,582
Polotovary	Pochutiny	47,455	51,475
Alkohol	Pecivo	42,875	50,148
Pecivo	Konzervy	45,547	50,0
Pochutiny	Polotovary	49,237	49,612
Pochutiny	Konzervy	45,547	49,162
Pecivo	Pochutiny	47,455	49,062
Polotovary	Konzervy	45,547	47,486
Konzervy	Pochutiny	47,455	47,185
Alkohol	Pochutiny	47,455	46,113
Mrazene_jidlo	Konzervy	45,547	45,531
Mrazene_jidlo	Pochutiny	47,455	45,04
Konzervy	Polotovary	49,237	43,928
Alkohol	Polotovary	49,237	43,152
Mrazene_jidlo	Polotovary	49,237	42,894

Obr. P.2.4 Model Carma

Jako poslední máme shrnutí modelu Carma, vypadá takto:



Příklady z dat STAGu

V této části přílohy budou popsány dva příklady, které byly vytvořeny za použití dat ze systému STAG Technické univerzity v Liberci. Aby nedošlo k porušení ochrany osobních údajů, je použita pouze struktura jednotlivých tabulek databáze studijní agendy. Data jsou pouze fiktivní. Nejdříve si popíšeme zdrojové soubory a následně datové streamy jednotlivých příkladů.

P.3 Udělení stipendia

Podle stipendijního řádu FM TUL se stipendium neposkytuje:

- v prvním roce studia
- studentům, kteří v předchozím akademickém roce studia nesplnili všechny studijní povinnosti určené studijním plánem nejpozději do data určeného harmonogramem TUL
- studentům, kteří studují další studijní program
- v době přerušení studia
- při překročení standardní doby studia

Stipendium se poskytuje studentům, jejichž vážený studijní průměr v uplynulém akademickém roce nepřekročí 1,8.

Naším úkolem bude vytvořit datový stream, který automaticky rozpozná, zda má student nárok na stipendium za vynikající studijní výsledky.

P.3.1 Data udělení stipendia

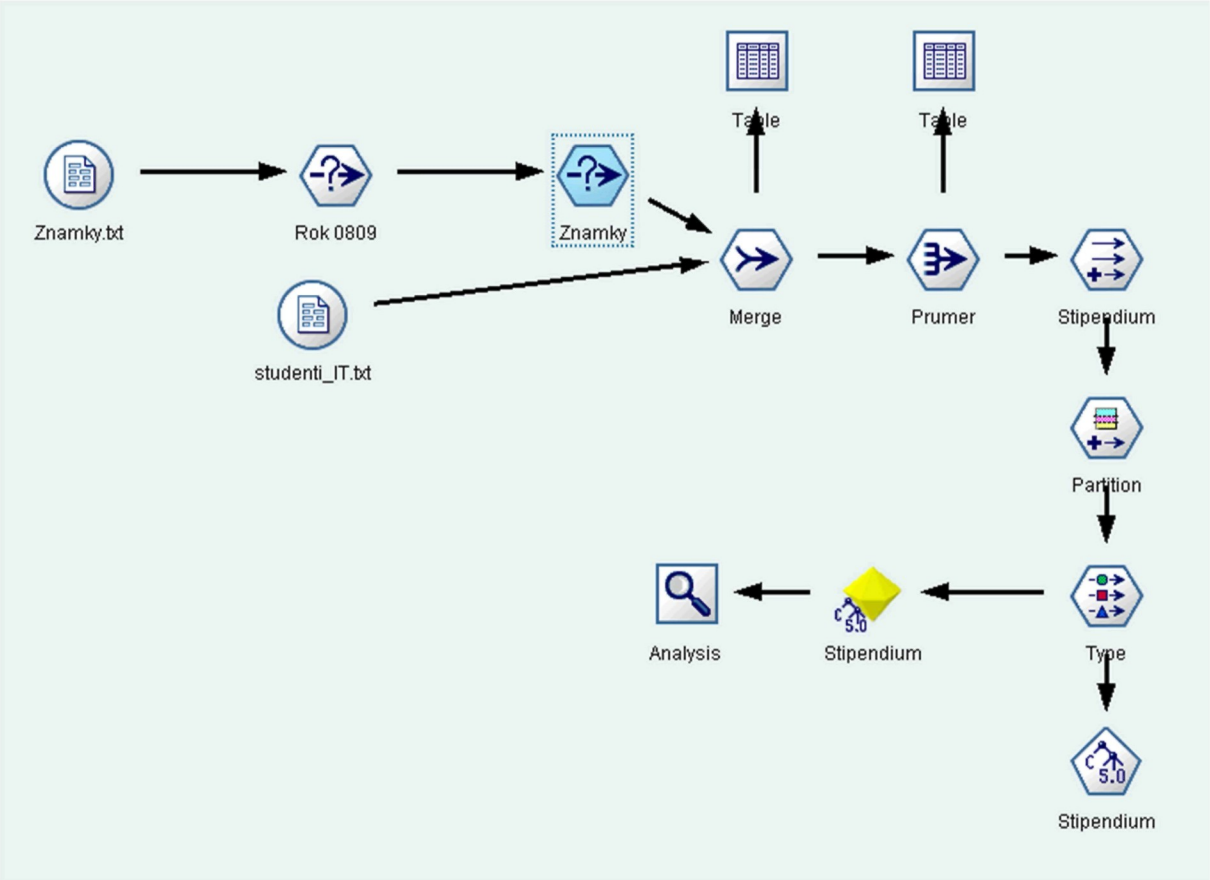
Máme k dispozici dva zdrojové soubory. Prvním je soubor *Znamky.txt*, jenž obsahuje následující pole:

Název	Popis
ID_OSOBY	Unikátní klíč k určení studenta
ID_PROGRAMU	Klíč studijního programu
PRAC_ZKR	Zkratka studijního programu
ZKR_PREDM	Zkratka předmětu
ROK_VARIANTY	Rok, kdy by předmět ohodnocen
ZNAMKA	Obdržená známka z předmětu
SLOVNE	Splnění zápočtu (S/N)
UZNANO	Uznání předmětu (A/N)

Druhým zdrojovým souborem je jedinečný seznam studentů s názvem *studenti_IT.txt*, který obsahuje jediné pole ID_OSOBY. V tomto poli jsou vypsaná ID studentů, kteří studují program Elektrotechnika a informatika.

P.3.2 Stream

Nyní si krok za krokem představíme postup při úpravě datové matice, aby z ní bylo možné rozhodnout, zda studentovi udělit stipendium. Hlavním cílem datového toku je vytvořit průměr u jednotlivého studenta. Na základě průměru pak bude rozhodnuto o stipendiu. Pro zjednodušení budeme pracovat pouze s daty za rok 2008 a 2009. Stream vypadá následovně



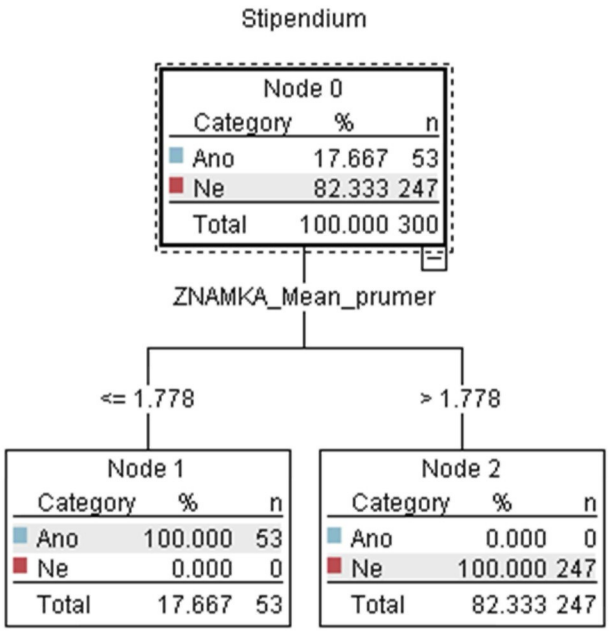
Obr. P.3.1 Stream stipendium studenta

Na obr. P.3.1 vidíme výše zmíněné zdrojové soubory *Znamky.txt* a *studenti_IT.txt*. Nyní si popíšeme funkci dalších uzlů:

- SELECT uzel *Rok0809* vybírá data za rok 2008 a 2009
- SELECT uzel *Znamky* ponechávám pouze číselné hodnoty v poli Znamka
- MERGE uzel plní dvě funkce v úpravě datové matice
    - Sloučení zdrojových souborů do jedné datové matice

- Filtruje pole, které dále nepotřebujeme k výpočtu průměru studenta – ID_PROGRAMU, ZKR_PREDM, SLOVNE, UZNANO

- AGGREGATE uzel *Prumer* používá jako klíč ID_OSOBY a k jednotlivým ID počítá průměr z pole Znamka
- Uzly TABLE jsou kontrolní tabulky k zjištění aktuálního stavu datové matice ve streamu
- DERIVE uzel *Stipendium* vytváří nové pole na základě následující podmínky: IF ZNAMKA_prumer (proměnná vytvořená v předchozím uzlu) < 1.8 THEN Ano ELSE Ne
- PARTITION uzel rozděluje datovou matici na trénovací a testovací množinu dat
- V datové matici jsou nyní 4 pole: ID_OSOBY, ZNAMKA_Prumer, Stipendium, Partition. V uzlu TYPE nastavíme tato pole na vstupní, případně výstupní, podle potřeby modelu C5.0. V tomto případě ID_OSOBY a ZNAMKA_prumer jsou nastaveny na vstupní a Stipendium na výstupní
- Model C5.0 nám na základě datové matice vytvoří model rozhodovacího stromu
- Samotný model si můžeme prohlédnout v uzlu *Stipendium* (viz. Obr P.3.2)
- Uzel ANALYSIS nám ukazuje hodnocení vytvořeného modelu (viz obr. P.3.3)



Obr. P.3.2 Model udělení stipendia

Results for output field Stipendium

Comparing $C-Stipendium with Stipendium

'Partition'	1_Training		2_Testing	
Correct	300	100%	284	100%
Wrong	0	0%	0	0%
Total	300		284	

Obr. P.3.3 Analýza modelu stipendia

P.4 Zapsané předměty

Druhý příklad, který byl vytvořen z umělých dat STAGu, využívá modelů asociačních pravidel. Zajímá nás pravděpodobnost současného zápisu dvou nebo více předmětů. Naším úkolem bude zjistit vazby mezi volitelnými předměty FM TUL.

P.4.1 Data zapsané předměty

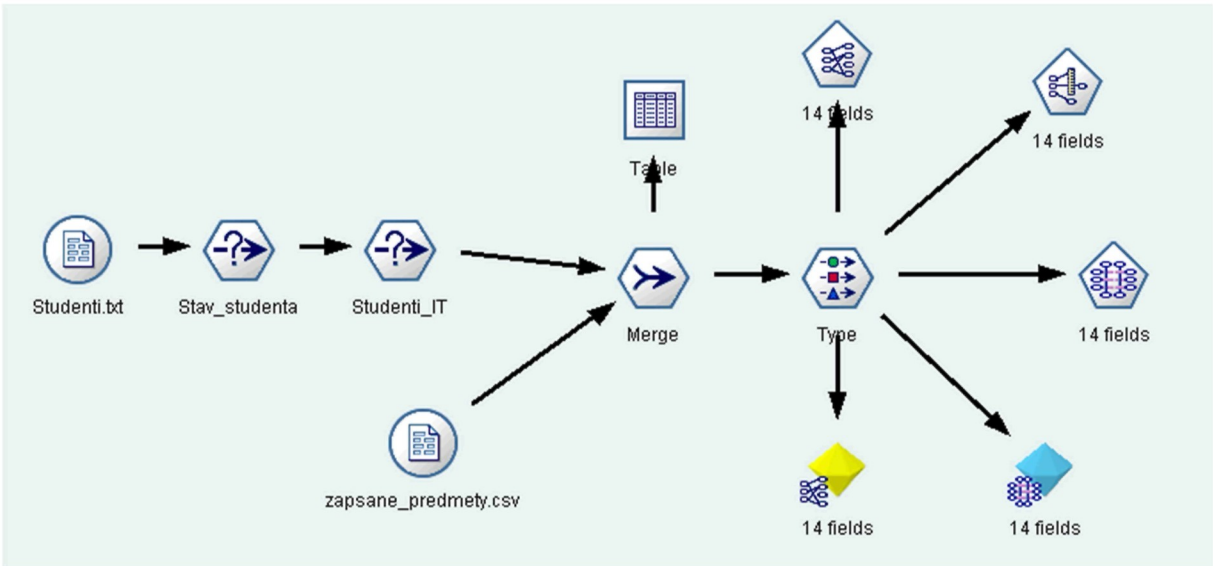
Zdrojové soubory pro tento příklad jsou opět dva. Soubor *studenti.txt* obsahuje tato pole:

Název	Popis
ID_OSOBY	Unikátní klíč studenta
ID_PROGRAMU	Číslo studijního programu
ID_OBORU	Číslo oboru
STAV	Stav studia (A,S,N)

Druhým zdrojovým souborem je soubor *zapsane_predmety.csv*, který obsahuje pole ID_OSOBY a další pole pojmenovaná zkratkami předmětů (např. PZR – počítačové zpracování řeči, PMR – Pokročilé metody zpracování řeči atd.). Tato pole obsahují binární hodnoty 0 nebo 1. Hodnota 1 znamená, že student má předmět zapsaný ve STAGu.

P.4.2 Stream

V této úloze nás zajímají zapsané předměty u studentů, kteří nejsou absolventi a v současné době opravdu studují (mají stav S). Datovou množinu dále omezíme na studenty, kterých se týkají výše zmíněné předměty, tedy obor Informační technologie. Výsledný stream bude vypadat takto:



Obr. P.4.1 Stream zapsané předměty

Na obr. P.3.4 vidíme výše popsané zdrojové soubory *Studenti.txt* a *zapsane_predmety.csv*. Nyní si opět popíšeme funkce zbylých uzlů:

- SELECT uzel *Stav_studenta* omezuje datovou množinu na studenty, kde Stav = S
- SELECT uzel *Studenti_IT* z množiny aktuálně studujících studentů dále vybírá pouze ty, kteří studují obor s ID 978 (Informační technologie)
- MERGE uzel plní dvě funkce:
    - Sloučení zdrojových souborů do jedné datové matice
    - Filtruje pole, které dále nepotřebujeme ke generování asociačních pravidel – ID_PROGRAMU, ID_OBORU, STAV
- TABLE uzel ukazuje výslednou datovou matici, ze které budou generována asociační pravidla
- V uzlu TYPE je nyní potřeba nastavit vstupní a výstupní pole podle potřeb jednotlivých modelů Apriori, GRI a CARMA. V tomto příkladu využijeme pouze směr *Both* (vstupně/výstupní) u všech polí označených zkratkou předmětu
- Následují tři uzly pro generování asociačních pravidel podle algoritmů Apriori, GRI a CARMA (shora)

P.4.3 Výsledek modelu Apriori

Uzel apriori při nastavení minimální podpory předpokladu 20%, minimální spolehlivosti pravidla 90% a maximálního počtu předpokladů 2 generuje následující pravidla:

Consequent	Antecedent	Support %	Confidence %
PVI	HSC ICP	20,29	100,0
NMP	KOT ELK	23,188	100,0
ICP	PVI TSW	27,536	100,0
PMR	MMP PZR	27,536	100,0
PZR	ELK PMR	30,435	95,238
PVI	ICP TSW	28,986	95,0

Obr. P.4.2 Model apriori

Celkem bylo vygenerováno 6 pravidel. Největší podporu má pravidlo ELK&PMR -> PZR. To znamená, že v datech se nejčastěji (z popsaných pravidel) vyskytuje trojice zapsaných předmětů ELK (Elektronika), PMR (Pokročilé metody zpracování řeči) a PZR (Počítačové zpracování řeči). Existuje 95 procentní šance, že když si studen zapíše první dva předměty ELK a PMR, zapíše si i PZR.

P.4.4 Výsledek modelu GRI

Nastavení uzlu GRI: minimální podpora 25%, minimální spolehlivost 65%, maximální počet předpokladů 3. Celkem vytvořených pravidel je 28. Následující obrázek ukazuje pouze část těchto pravidel. Model GRI generuje tato pravidla:

Consequent	Antecedent	Support %	Confidence %
ICP	PVI TSW	27,54	100,0
PMR	PZR MMP	27,54	100,0
PVI	ICP TSW	28,99	95,0
PZR	MMP PMR	30,43	90,48
PMR	PZR	55,07	89,47
ELK	KOT NMP	26,09	88,89
PZR	PMR	56,52	87,18
KOT	MAR	30,43	85,71
TSW	KOT NMP	26,09	83,33
MAR	KOT	31,88	81,82
NMP	KOT	31,88	81,82
TSW	NMP	36,23	80,0
ICP	PVI	47,83	78,79
MMP	MAR	30,43	76,19
TSW	ICP PVI	37,68	73,08
ELK	KOT	31,88	72,73
TSW	KOT	31,88	72,73
NMP	PZR TSW	26,09	72,22
PVI	ICP	52,17	72,22

Obr. P.4.3 Model GRI

Z tohoto modelu jsou patrné některé dvojice předmětů, které si studenti zapisují nejčastěji společně. Např. opět PZR -> PMR s podporou 55% a spolehlivosti téměř 89,5%. Další poměrně často vyskytující se dvojice je ICP (Interakce člověka s počítačem) -> PVI (Počítačové vidění) s podporou 52,17% a spolehlivostí 72,22%.

P.4.5 Výsledek modelu CARMA

Model CARMA vygeneroval největší počet asociačních pravidel, celkem 138. Na následujícím obrázku vidíme vybraná pravidla s největší spolehlivostí. Nastavení uzlu CARMA: min. podpora 20%, min. spolehlivost 20%, maximální velikost pravidla 10. Pravidlo PZR->PMR má stejné hodnoty jako u modelu GRI. Ze všech tří modelů je patrné, že nejčastější dvojice zapisovaných předmětů je PZR a PMR.

Consequent	Antecedent	Support %	Confidence %
PMR	PZR MMP	27,536	100,0
ICP	PVI TSW	27,536	100,0
NMP	KOT ELK	23,188	100,0
PVI	HSC ICP	20,29	100,0
PZR	PMR ELK	30,435	95,238
PVI	ICP TSW	28,986	95,0
PMR	PZR SIP	26,087	94,444
KOT	ELK NMP	24,638	94,118
NMP	KOT TSW	23,188	93,75
KOT	MAR NMP	21,739	93,333
NMP	ELK TSW	21,739	93,333
PMR	PZR ELK	31,884	90,909
PZR	MMP PMR	30,435	90,476
PMR	PZR	55,072	89,474

Obr. P.4.4 Model CARMA

P.5 Monitoring zkušebního provozu stroje

Příklad, ve kterém využijeme modelu neuronové sítě, se týká monitoringu zkušebního provozu přístroje.

P.5.1 Úloha

Zavádění nových technologií do výroby je spojeno s určitými riziky. Nové přístroje se především testují na selhání. Firma chce do výroby zavést nový přístroj a rozhodla se ho prozkoušet. Nějakou dobu zaznamenávala různé údaje o jeho provozu, chybová hlášení a jeho opravdový stav. Na základě těchto dat chce predikovat, za jakých podmínek dojde k poruše přístroje.

Datový zdroj obsahuje několik časových řad záznamů o stavu stroje. Každý záznam je složen z těchto informací:

Název	Popis
Pořadové číslo	Čas, kdy byl proveden záznam
Výkon	Výkon stroje
Teplota	Teplota stroje
Tlak	0 pro normální stav, 1 pro výstražný stav tlaku
Čas	Čas od poslední kontroly
Status	0 normální stav, 101, 202, 303 chybové stavy
Výsledek	Stav do kterého systém dojde (0, 101, 202, 303)

Hlavní částí tohoto příkladu bude fáze přípravy dat do podoby potřebné pro jednotlivé modely. Více v popisu streamu příkladu.

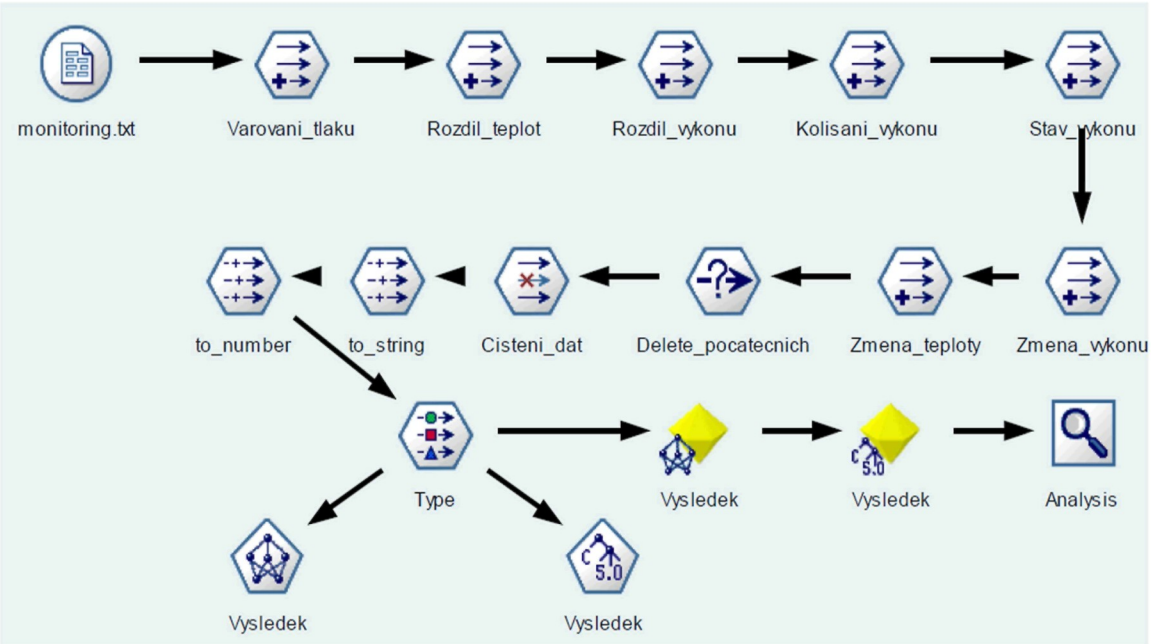
P.5.2 Stream

- DERIVE *Varovani_tlaku* sčítá počet výstražných stavů tlaku pro jednotlivé časové řady zvlášť
- *Rozdil_teplot* počítá rozdíly teplot ve dvou následujících záznamech
- *Rozdil_vykonu* analogicky s předchozím počítá rozdíly výkonu
- *Kolisani_vykonu(dále jako kv)* vrací hodnotu TRUE, pokud změna výkonu v tomto a předchozím záznamu byla opačná[1]

[1] Z kladné na zápornou nebo opačně

- *Stav_vykonu* reaguje na předchozí stav implicitně vrací hodnotu Stabilni. Pokud dvě po sobě jdoucí hodnoty proměnné Kolisani_vykonu jsou TRUE, změní hodnotu na Kolisave. Do stavu Stabilni se vrací, pokud hodnoty proměnné Kolisani_vykonu jsou FALSE v posledních pěti záznamech.
- *Zmena_vykonu* počítá průměr z posledních pěti hodnot proměnné *Rozdil_vykonu*
- *Zmena_teploty* analogicky s proměnnou *Rozdil_teplot*
- Uzel SELECT s názvem *Delete_pocatecnich* smaže první záznam z každé časové řady, protože odvozené diferenční charakteristiky v nich nejsou definované
- Uzel FILTER *Cisteni_dat* odfiltruje všechny proměnné kromě Cas, Status, Vysledek, Varovani_tlaku, Stav_vykonu, Zmena_vykonu a Zmena_teploty
- Uzel FILLER *to_string* převádí číselné hodnoty proměnné *Vysledek* na řetězce
- *to_number* převádí hodnoty proměnné Varovani_tlaku na integer
- V uzlu TYPE, nastavíme proměnnou *Vysledek* jako výstupní hodnotu (Direction OUT) a ostatní proměnné jako vstupní (Direction IN)
- Na uzel TYPE napojíme dva modely, C5.0 a Neural Net

Výsledný stream vypadá následovně:



Obr. P.5.1 Stream monitoringu zkušebního provozu

Po spuštění celého streamu se vytvoří dva modely. Rozhodovací strom C5.0 a Neuronová síť. Tyto modely (ikonky diamantu) vložíme do proudu a otestujeme si je pomocí výstupního uzlu Analysis. Výsledkem jsou dvě tabulky, ve kterých je srovnání výstupů jednotlivých modelů s modelovanou proměnnou *Vysledek* a porovnání výsledků rozhodovacího stromu a neuronové sítě. Tyto tabulky

si můžeme prohlédnout na následujícím obrázku. V nich vidíme, že výstup rozhodovacího stromu se v 99,79% shoduje s cílovou proměnnou Vysledek. Neuronová síť dosáhla prakticky stejného výsledku s 99,75%.

Results for output field Vysledek

Individual Models

Comparing $N-Vysledek with Vysledek

Correct	2 377	99,75%
Wrong	6	0,25%
Total	2 383	

Comparing $C-Vysledek with Vysledek

Correct	2 378	99,79%
Wrong	5	0,21%
Total	2 383	

Agreement between $N-Vysledek $C-Vysledek

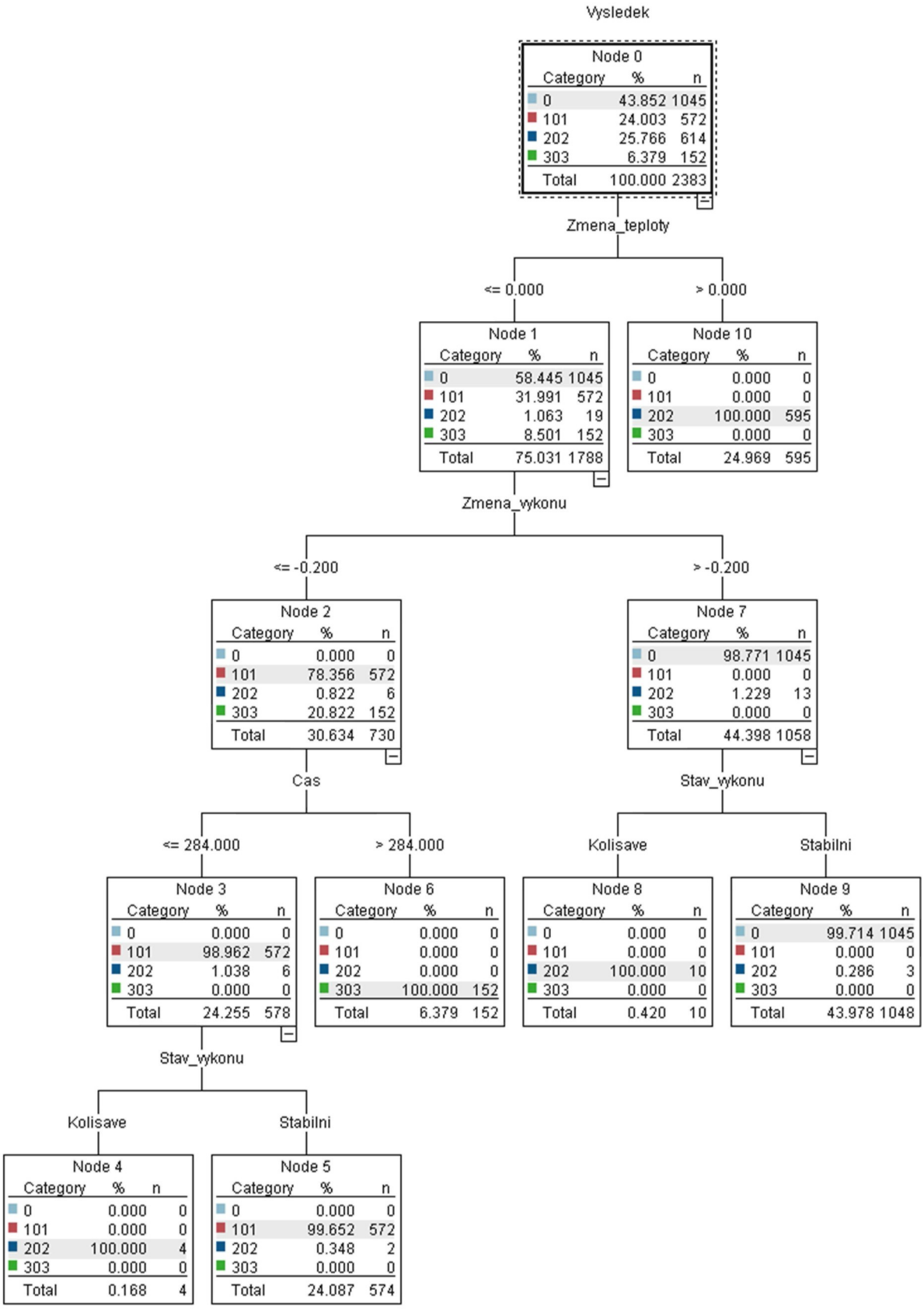
Agree	2 382	99,96%
Disagree	1	0,04%
Total	2 383	

Comparing Agreement with Vysledek

Correct	2 377	99,79%
Wrong	5	0,21%
Total	2 382	

Obr. P.5.2 Analýza modelů

Výhodou rozhodovacího stromu je jeho jednoduchá interpretace. Prohlížením modelu na záložce Viewer je tento strom zobrazený (viz. obr. P.5.3). Závěr, který můžeme interpretovat na jeho základě je ten, že vzrůstající teplota je stoprocentní indikátor chybového stavu 202. Chybový stav 303 nastává, pokud teplota neroste (Zmena_teploty <= 0), výkon výrazně klesá (Zmena_vykonu <= -0,2), poslední kontrola byla provedena před 284 dny (Cas <= 284) a výkon je stabilní (Stav_vykonu = Stabilni).



Obr. P.5.3 Rozhodovací strom monitoringu stroje