TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

KLASIFIKAČNÍ ALGORITMY V DATAMININGOVÝCH ÚLOHÁCH

Bakalářská práce

Studijní program: B2646 - Informační technologie
Studijní obor: 1802R007 - Informační technologie

Autor práce: **Petr Franz**
Vedoucí práce: RNDr. Klára Císařová, Ph.D.

Květen 2015

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

CLASSIFICATION ALGORITHMS IN DATAMINING

Bachelor Thesis

Study programme: B2646 - Information Technology
Study branch: 1802R007 - Information Technology

Author: **Petr Franz**
Supervisor: RNDr. Klára Císařová, Ph.D.

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2014/2015**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Petr Franz**
Osobní číslo:	**M12000127**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Klasifikační algoritmy v dataminingových úlohách**
Zadávající katedra:	**Ústav mechatroniky a technické informatiky**

Zásady pro vypracování:

1. Seznamte se s dataminingem a nástrojem IBM SPSS Modeler.
2. Prostudujte klasifikační algoritmy používané v dataminingových úlohách.
3. Vybraný algoritmus, podle pokynů vedoucí, nastudujte.
4. Navrhněte výkladový způsob vizualizace vybraného algoritmu pro výuku DM a naprogramujte jej.
5. Téma zpracujte pro e-learningový kurz na portále ALS.

Rozsah grafických prací: **dle potřeby dokumentace**

Rozsah pracovní zprávy: **30–40 stran**

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

[1] **Yong Yin, Ikou Kaku, Jiafu Tang: Data Mining, Springer London Ltd, 2011.**

[2] **Steve McConnell: Dokonalý kód, Computer Press, 2006.**

[3] **Kotler Philip: Marketing management, Grada, 2003.**

[4] **Hendl J.: Přehled statistických metod zpracování dat, Portál, 2006.**

[5] **Olivia Parr Rud: Datamining, Computer Press, 2006.**

[6] **http://www.msps.cz/data-mining/**

[7] **http://www.spss.cz/pasw_modeler.htm**

[8] **tutorialy k SAP a další materiály na WWW stránkách.**

Vedoucí bakalářské práce: **RNDr. Klára Císařová, Ph.D.**
Ústav mechatroniky a technické informatiky

Datum zadání bakalářské práce: **10. října 2014**

Termín odevzdání bakalářské práce: **15. května 2015**

prof. Ing. Václav Kopecký, CSc.
děkan

L.S.

doc. Ing. Milan Kolář, CSc.
vedoucí ústavu

V Liberci dne 10. října 2014

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, zejména § 60 - školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím bakalářské práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum: 14. 5. 2015

Podpis: Franc

Poděkování

Rád bych na tomto místě poděkoval vedoucí práce, paní RNDr. Kláře Císařové, Ph.D. a všem ostatním, kteří se na ní podíleli, za poskytnutý čas, ochotu a trpělivost během konzultací. Také mé rodině za jejich veškerou podporu.

Abstrakt

Tato práce seznamuje čtenáře se základními principy dataminingu a popisuje klasifikační algoritmy, jež jsou využívány při řešení různých reálných problémů. Vybrané algoritmy jsou pak demonstrovány na konkrétních datech.

Práce dále popisuje vznik desktopové aplikace pro vizualizaci vybraných algoritmů, jakož to nástroje použitelného pro výuku.

Klíčová slova

Datamining, rozhodovací stromy, informační zisk, entropie, prediktor, predikovaný atribut.

Abstract

This thesis introduces the basic priciples from datamining and describes classification algorithms, which are used in solving different real problems. Selected classification algorithms are then demonstrated on specific data.

This thesis also describes the emergence of desktop-application for visualisation of selected classification algorithms, as a tool, which is usefull for teaching.

Keywords

Datamining, decision trees, information gain, entropy, predictor, predicted attribute.

Obsah

1 Úvod

Bakalářská práce je rozdělena do dvou částí. V první teoretické části se bakalářská práce zaměřuje na zavedení pojmů, seznámení se s Dataminingem a procesem dobývání znalostí z databází, dále pak představuje rozhodovací stromy, a s nimi spojené různé problémy, jako je například problém chybějících hodnot a podobně, a klasifikační algoritmy, které se používají pro tvorbu rozhodovacího stromu. Konkrétně se jedná o algoritmy Entropie, Informační zisk a GINI index.

V druhé praktické části je popsána tvorba aplikace, která má sloužit k zobrazení vybraných klasifikačních algoritmů nad konkrétními testovacími daty. Do aplikace byly implementovány klasifikační algoritmy Entropie a Informační zisk. Jsou zde popsány stěžejní části programu, hlavně pak vzniklé důležité třídy a metody. Například metody pro výpočty prováděné v rámci klasifikačních algoritmů, metody pro tvorbu stromu a práce s daty a jejich zobrazení. Čtenář v této části najde vývojové diagramy některých metod a tabulky znázorňující jak klasifikační algoritmus pracuje s daty. V neposlední řadě seznamuje s realizací uživatelského prostředí.

2 Cíl práce

Hlavním cílem bakalářské práce bylo vytvořit program, který by vizualizoval vybrané klasifikační algoritmy používané v Dataminingu. V našem případě se jedná o algoritmy Informačního zisku a Podmíněné entropie. Tedy oba algoritmy vizualizovat po jednotlivých krocích na připravených datech a uvádět výsledky výpočtů. Program má sloužit jako nástroj pro výuku.

3 Úvod do problematiky dobývání znalostí z databází

Pojem Datamining se objevuje v historii už na počátku devadesátých let minulého století, a to v Americe, na konferencích o umělé inteligenci. Jednoduše řečeno se jedná o metodu, která dokáže v rozsáhlých datech vyhledávat informace a následně modelovat a analyzovat závislosti v těchto datech. Od té doby se mluví o Dobývání znalostí z databází (Knowledge discovery in databases) nebo také dobývání znalostí a dolování z dat, tedy data mining.

Tato metoda, na rozdíl od předchozích používaných statistických metod, využívá určitou předpřípravu dat pro analýzu a konečnou interpretaci výsledných znalostí. K tomu nám může sloužit komerční program od společnosti IBM, IBM SPSS Modeler, který umožňuje nejen kontrolu dat z hlediska chybějících hodnot, ale také například poskytuje informace o setřiditelnosti a dalších možnostech přípravy dat. V neposlední řadě obsahuje mnoho klasifikačních algoritmů, jejichž výsledkem spuštění nad připravenými daty jsou zobrazené informace v souvislostech.

Celý proces dobývání znalostí z databází se skládá z pěti po sobě jdoucích částí. Selekce, předzpracování, transformace, dolování, interpretace. Jednotlivé kroky nám říkají, že z velkého množství výchozích dat v databázích selekcí vybereme pro nás zajímavá data, ty zpracujeme a následně je transformujeme do vhodné podoby pro daný algoritmus dolování. Zobrazené výsledky projdou interpretací a můžeme jim tedy říkat znalosti.

Kroky selekce, předzpracování a transformace můžeme shrnout do procesu získání všech dostupných dat použitelných pro řešení zadaného problému. V tomto procesu probíhá veškerá analýza dat. S tím souvisí výběr analytických metod, kterých je celá řada a často se kombinují. Zjišťuje se, zda jsou data opravdu odpovídající k danému problému a zda nejsou potřeba další doplňující data z externího zdroje. Na konec se data doplňují o chybějící hodnoty, případně se odstraňují odlehlá data. Tím jsou data připravená na následující dolování, tedy data mining.

Při dolování se nad daty spouští vybrané analytické metody, jež mezi daty vyhledávají zajímavé vztahy. Jednotlivé metody mohou být spuštěny vícekrát v závislosti na výsledcích, přičemž se výstup stává novým vstupem do metody.

Po skončení dolování přichází na řadu interpretace. Ta slouží hlavně ke srozumitelnému zpracování výsledků získaných z jednotlivých analytických metod. Je tedy nezbytná pro konečné získání znalostí. Výstupem interpretace obvykle bývá analytická zpráva, data, která lze použít přímo, nebo také spuštění jiné aplikace.

Bakalářská práce se hlavně zaměřuje na krok dolování a s ním vybrané analytické metody a klasifikační algoritmy.

4 Dolování z dat, Datamining

Jak již bylo zmíněno, pro dolování je využíváno mnoho různých analytických metod. Mezi nejčastěji používané analytické metody patří metody pro tvorbu rozhodovacích stromů.

4.1 Rozhodovací stromy

Rozhodovací stromy se mohou požívat jako nejrůznější klíče, například k určování živočichů a rostlin v biologii. Patří mezi nejznámější klasifikační algoritmy. Pracují na principu „rozděl a panuj“, kdy se data postupně rozpadají na malé podmnožiny, respektive uzly, a to tak, aby v jedné podmnožině byly pouze příklady jedné třídy. Při spuštění algoritmu máme tedy pouze jednu množinu, tzv. hlavní uzel nebo také kořen, a při skončení dostáváme na výstupu podmnožiny, které jsou tvořené příklady stejné třídy. Jde o metodu přístupu shora dolů.

Obecné schéma můžeme zapsat následovně:

1) Zvolte jeden atribut jako kořen stromu.
2) Rozdělte data v tomto uzlu na podmnožiny podle hodnot zvoleného atributu a přidejte uzel pro každou podmnožinu.
3) Existuje-li uzel, pro který nepatří všechna data do stejné střídy, tak opakujte pro tento uzel postup od bodu 1, jinak skončí.

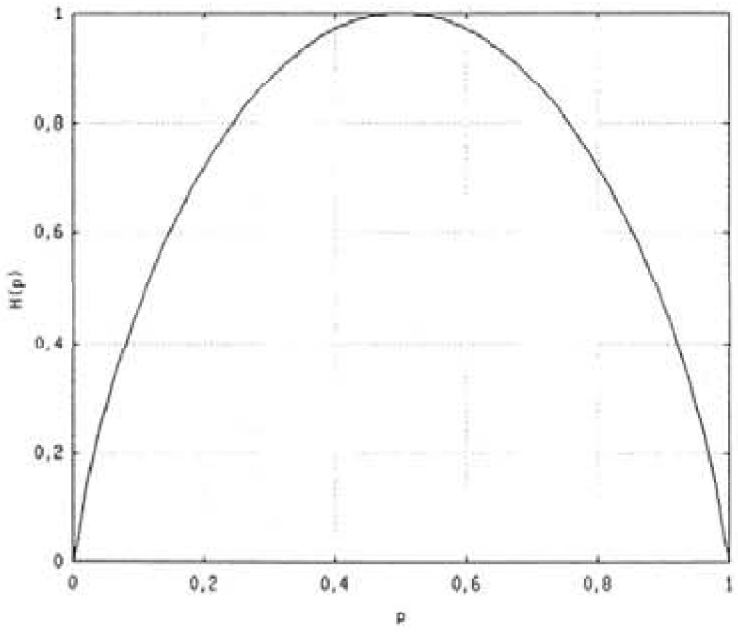
Algoritmus pracuje pro kategoriální data. To znamená, že počet uzlů vytvořených v kroku 2 odpovídá počtu hodnot daného atributu. Volba kořenového atributu v kroku 1 je založena na charakteristikách atributu převzaté z teorie informace a pravděpodobnosti. Jedná se o *entropii*, *informační zisk* a *poměrný informační zisk* nebo *Gini index*.

4.2 Entropie

V přírodních vědách udává pojem entropie míru neuspořádanosti nějakého systému. V teorii informace je definována funkcí

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

Kde p_t udává pravděpodobnost výskytu třídy t. Jde o relativní četnost, která je počítaná na určité množině. T pak udává celkový počet tříd.



Obrázek 4.2 Entropie

Na obrázku 4.2 je vidět graf průběhu entropie v závislosti na pravděpodobnosti p v případě dvou tříd. Je-li p jedna a tedy všechny příklady patří do této třídy nebo je-li p nula a tedy žádný příklad nepatří do této třídy, pak je entropie nulová. V případě, že je p rovno jedné polovině, a to znamená, že jsou obě třídy zastoupeny stejným počtem příkladů, je entropie maximální.

4.2.1 Obecný výpočet entropie

Pro každou třídu *v*, kterou nabývá atribut *A*, je podle vzorce na skupině příkladů spočítána entropie $H(A_{(v)})$. Skupina příkladů je pokrytá kategorií $A_{(v)}$.

$$H(A(v)) = -\sum_{t=1}^{T} \frac{n_t(A(v))}{n(A(v))} * \log_2 \frac{n_t(A(v))}{n(A(v))}.$$

Následně je spočítána střední entropie atributu *A* jako vážený součet entropií $H(A_{(v)})$, kde váhy v součtu jsou četnosti kategorií $A_{(v)}$ v datech.

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

Pro větvení stromu je pak vybrán atribut s nejmenší entropií $H(A)$.

4.3 Informační zisk, poměrný informační zisk

Informační zisk je odvozen od entropie. Jde o rozdíl entropie predikovaného neboli cílového atributu *H(C) a uvažovaného* atributu, též prediktora, *H(A)*. Je tak měřena redukce entropie, která je způsobena volbou atributu *A*.

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

$$H(C) = -\sum_{t=1}^{T} \frac{n_t}{n} * \log_2 \frac{n_t}{n}.$$

Na rozdíl od entropie, ale informační zisk hledá atribut s maximální hodnotou. Je to dáno tím, že entropie pro celá data není závislá na atributu. Z toho plyne, že první člen rozdílu je konstantní a tedy maximální rozdíl nastane tehdy, pokud druhý člen rozdílu bude minimální.

Nevýhodou ale je, že se nezahrnuje do úvahy počet hodnot daného atributu. Jde pouze o odlišení příkladů různých tříd na základě vybraného atributu. Problém nastane například v případě, že bychom pro další větvení jako atribut vybrali pořadové číslo příkladu. Tento atribut by sice umožnil bezchybně rozdělit a klasifikovat data, ale byl by zcela nepoužitelný pro klasifikaci dalších příkladů. Z toho důvodu byl zaveden *poměrný informační zisk*.

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

Větvení(A) je vlastně entropie dat k hodnotám atributu *A*.

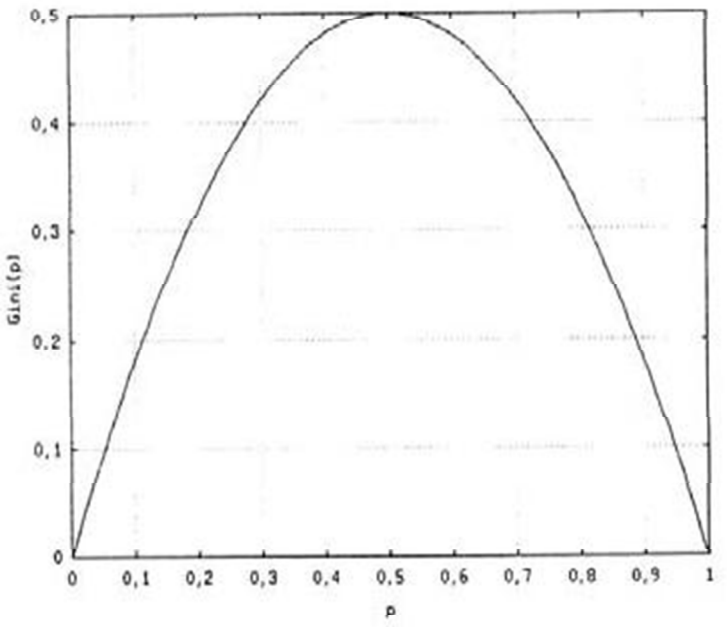
$$Větvení(A) = -\sum_{v \in Val(A)} \frac{n(A(v))}{n} \log_2 \frac{n(A(v))}{n}.$$

4.4 GINI index

GINI index v podstatě představuje stejnou roli jako entropie. Index se spočítá pomocí vzorce:

$$GINI = 1 - \sum_{t=1}^{T} p_t^2.$$

Hodnota p_t zde opět znamená počet příkladů t-té třídy zjišťovaný na nějaké množině.



Obrázek 4.4 GINI index

Na obrázku 4.4 je vidět graf závislosti GINI indexu na pravděpodobnosti jedné ze dvou tříd. Hodnota indexu je opět minimální v případě, že příklady patří do jedné třídy a maximální pokud jsou rovnoměrně rozděleny mezi obě třídy.

Hodnota indexu se počítá stejně jako hodnoty u entropie. Pro každý atribut se spočítá vážený součet pro jednotlivé hodnoty atributu, kdy váhy jsou opět relativní četnosti daných hodnot.

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

A také stejně jako u entropie je vybrán pro další větvení atribut s nejmenší hodnotou.

Další možností jak pracovat s GINI indexem je schopnost maximalizovat rozdíl mezi predikovaným atributem a prediktorem, stejně jako je to u informačního zisku.

$GINI(C) - GINI(A)$, kde $GINI(C) = 1 - \sum_{t=1}^{T}(\frac{n_t}{n})^2$.

4.5 Prořezávání stromů

Při spuštění klasifikačních algoritmů nad trénovacími daty dochází k bezchybnému rozdělení. To však, ale ve většině případů není možné nad reálnými daty. V tom případě může docházet k přeučení a navíc může být výsledný strom příliš košatý a tedy málo srozumitelný. Kvůli tomu se ve většině algoritmů požaduje, aby v podmnožině, tedy v listech stromu, vždy převažovaly příklady jedné třídy.

Po prořezání bývá strom menší a srozumitelnější. Ovšem může docházet k chybám. Prořezávání lze provádět dvěma způsoby:

- Modifikací původního algoritmu.
- Následným prořezáním úplného stromu.

Vzhledem k tomu, že je obtížné poznat, kdy přesně ukončit růst stromu, bývá praktičtější druhý způsob. V něm se vytvoří nejdříve úplný strom a následně se pro uzly, od nichž se odvíjí další podstrom, rozhoduje, jak moc jejich náhrada listem zhorší úplný strom. Náhrada takového uzlu znamená, že všechny příklady uzlu budou shrnuty do stejné třídy.

Hlavním problémem je to, jak poznat, jestli lze uzel nahradit. K tomu se mohou využívat buď nová validační data a ta se použijí pro testování redukce, nebo je redukce odhadována pomocí statistických testů z původních dat. Ne vždy ale bude strom klasifikovat příklady bezchybně.

4.6 Chybějící hodnoty a ceny atributů

4.6.1 Chybějící hodnoty

Při práci s reálnými daty je velmi pravděpodobné, že některá data budou chybět. To lze řešit v kroku předzpracování, ale také v některých algoritmech pro tvorbu stromů.

Jsou různé způsoby, jak se s tím algoritmus vypořádá. Například na místo chybějící hodnoty atributu nastaví nejčastější hodnotu stejného atributu, nebo je možné spočítat relativní četnosti všech hodnot atributu a chybějící hodnotě nastavit hodnoty s váhami danými relativními četnostmi.

4.6.2 Ceny atributů

V některých případech je nutné vědět, jak velká je cena za získání hodnoty nějakého atributu. Cena se pak může aplikovat už při růstu stromu. U informačního zisku je v algoritmu cena atributu například využita jako dělitel spočteného zisku atributu daného na čtverec.

$\frac{Zisk(A)^2}{Cena(A)}$ Cena(A) jsou zde náklady za zjištění hodnoty atributu *A*.

4.7 Použití rozhodovacích stromů

Při používání rozhodovacích stromů ke klasifikaci jsou kladeny otázky s odpovědí typu ano/ne. Převedeno do praxe se díky těmto odpovědím lze postupně propracovat od kořenového uzlu hlouběji až ke konečnému listovému uzlu, který splňuje požadavek zařazení příkladu do třídy.

Vhodné úlohy pro rozhodovací stromy:

- Klasifikace příkladů do malého počtu tříd.
- Data jsou zatížena šumem.
- Data mohou obsahovat chybějící hodnoty.
- Koncept je tvořen disjunkcemi.
- Příklady jsou dané hodnotami atributů.

5 Programování algoritmů a tvorba programu

Hlavním cílem bakalářské práce bylo vytvořit program pro vizualizaci vybraných klasifikačních algoritmů na trénovacích datech a zobrazit uživateli výsledný strom i jednotlivé kroky a výsledky výpočtů.

Byla proto navržena desktopová aplikace psaná v objektovém jazyce C#. K tomu bylo využito vývojové prostředí Visual Studio 2012.

5.1 Vstup

Vstupem do aplikace je externí soubor **.csv* nebo **.txt*. Soubor obsahuje tabulková data s hodnotami, které jsou od sebe separovány jednoduchým oddělovačem pro daný typ souboru.

Klient	Příjem	Konto	Pohlaví	Nezaměstnaný	Úvěr
1	vysoký	vysoké	žena	ne	ANO
2	vysoký	vysoké	muž	ne	ANO
3	nízký	nízké	muž	ne	NE
4	nízký	vysoké	žena	ano	ANO
5	nízký	vysoké	muž	ano	ANO
6	nízký	nízké	žena	ano	NE
7	vysoký	nízké	muž	ne	ANO
8	vysoký	nízké	žena	ano	ANO
9	nízký	střední	muž	ano	NE
10	vysoký	střední	žena	ne	ANO
11	nízký	střední	žena	ano	NE
12	nízký	střední	muž	ne	ANO

Tabulka 5.1 Vstupní data

5.1.1 Načtení dat

V abstraktní třídě Open byla vytvořena metoda open(). Zde byl vytvořen Streamreader, který v cyklu přečte řádek souboru, ten následně rozdělí podle daného oddělovače (ten je určen potomkem třídy Open – TxtOpen, CSVOpen) a jednotlivá vzniklá slova uloží do tabulky (nejprve názvy sloupců, a pak hodnoty řádků), ketou metoda ukládá do proměnné typu List<>.

5.2 Kategorizace vstupních dat

V programu byla vytvořena třída Kategorie. Ta pracuje na základě slovníku, tedy klíče a hodnoty. Třída Kategorie vlastně představuje jednotlivý atribut. Je v ní uchováno jméno atributu a do slovníku se ukládají názvy tříd a jejich četnosti. Dále pak může uchovávat přímo jméno třídy.

Třída Kategorie je pak v hlavní části programu reprezentována polem o velikosti počtu atributů a je nastavována na dvou místech. Na prvním počítá obyčejné četnosti bez nějakých dalších závislostí a na druhém počítá četnosti v závislosti na predikovaném atributu. Obyčejné četnosti pak zobrazuje v tabulkách pro jednotlivé atributy (Tabulka 5.2).

Příjem	Počet
Vysoký	5
Nízký	7

Tabulka 5.2 Zobrazení četností tříd pro Příjem

Takto spočítané četnosti jsou následně využívány v algoritmech Entropie a Informačního zisku.

5.2.1 Čítání obyčejných četností

Pole je zde inicializováno jako jednorozměrné na velikost počtu atributů a v cyklu je každému prvku přiřazeno jméno atributu.

Následně se cyklicky projde celá aktuální tabulka po buňkách v řádku a porovnává se hodnota v řádku s hodnotou uloženou na daném místě v poli kategorií. Pokud takový prvek již existuje, zvýší se pouze jeho četnost o jedna. Pokud ne, vytvoří se v na daném místě v poli a tedy v jeho slovníku další záznam o četnosti jedna.

Výsledek čítání je vidět na příkladu v tabulce: Tabulka 5.2 Zobrazení četností tříd pro Příjem.

5.2.2 Čítání četností v závislosti na predikovaném atributu

Tato metoda již počítá s uživatelovým nastavením cílového neboli predikovaného atributu na rozdíl od předchozí metody, která prostě čítá bez nějaké zvláštní podmínky.

Jde tedy o dost složitější proces. To je vidět už vlastně na vytvoření nového pole Kategorií. Už nejde pouze o jednorozměrné pole, ale o dvourozměrné. Jako první rozměr je opět počet atributů tabulky. Jako druhý je ale počet tříd daného atributu. Počet tříd je získán z délky slovníku obyčejných četností, které se nastavují automaticky. Průchodem tohoto konkrétního slovníku pro konkrétní atribut je nastaveno položce v poli jméno atributu a jméno třídy.

Pro čítání četností predikovaného atributu pro danou třídu jsou pak využity tři do sebe vnořené cykly. V nich se prochází po buňkách hodnoty v aktuální tabulce, s tím, že se vynechává sloupeček predikovaného atributu, a porovnává se aktuální hodnota buňky se jménem procházející třídy. Pokud se neshodují, tak se algoritmus posouvá na další buňku. Pokud je zde shoda, dochází ke konečnému rozhodování, jako tomu bylo u předchozí metody. Tedy zda hodnota v řádku sloupečku predikovaného atributu již byla přidána do slovníku pro danou třídu či nebyla.

Následující tabulky (Tabulka 5.2-1 Konkrétní hodnoty a Tabulka 5.2-2 Četnosti v závisloti na Úvěru) algoritmus znázorňují pro prediktora *Příjem* a predikovaným atributem je *Úvěr*:

Příjem	Úvěr
vysoký	ANO
vysoký	ANO
nízký	NE
nízký	ANO
nízký	ANO
nízký	NE
vysoký	ANO
vysoký	ANO
nízký	NE
vysoký	ANO
nízký	NE
nízký	ANO

Tabulka 5.2-1 Konkrétní hodnoty

Příjem/Úvěr	Ano	Ne
Vysoký	5	0
Nízký	3	4

Tabulka 5.2-2 Četnosti v závisloti na Úvěru

Ve slovníku jsou tedy uloženy hodnoty prediktora *Příjem* pro třídu *Vysoký*: ANO, 5.

5.3 Příprava algoritmů

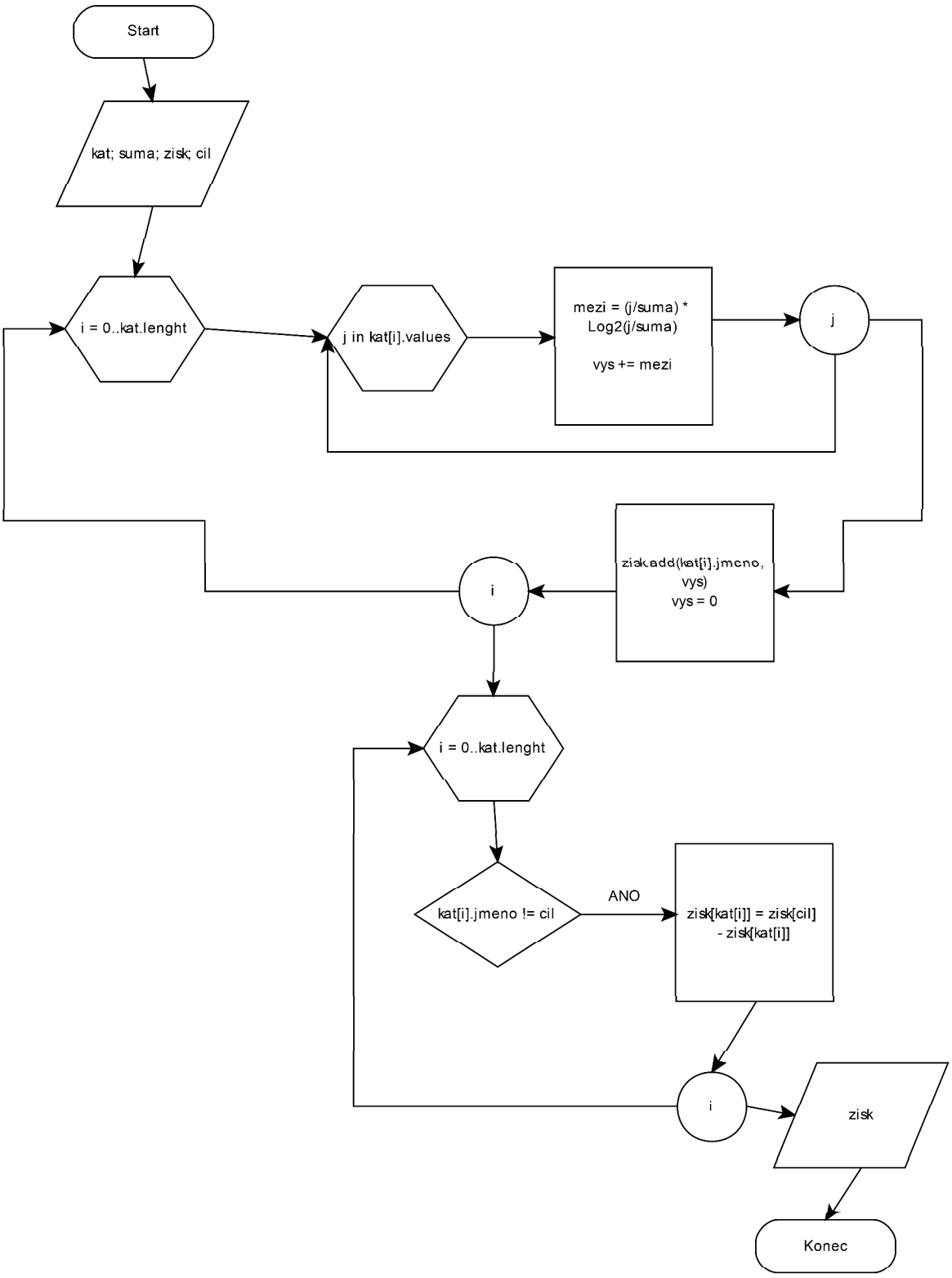
Bakalářská práce se zaměřuje na dva algoritmy. Algoritmus Informační zisk a algoritmus Podmíněná entropie. Oba algoritmy byly popsány v samostatných kapitolách: 4.3, 4.2. Tato kapitola se tedy zaměřuje na programové zpracování.

Vzhledem k tomu, že oba algoritmy jsou založeny na počítání s četnostmi kategorií, mají tedy podobné vstupní hodnoty, a oba vracejí nějaká spočítaná čísla s konkrétním pojmenováním, byla vytvořena abstraktní třída *AData*, která obsahuje oba typy četností popsané v předchozí kapitole (5.2), tedy dvě proměnné které reprezentují pole Kategorie. Třída pak obsahuje proměnnou hodnotu *suma*, která je určena počtem řádků aktuální tabulky. Výsledky výpočtů jsou zapsány do proměnné typu slovníku. Nakonec třída obsahuje dvě abstraktní metody pro *výpočet()*.

5.3.1 Metoda pro Informační zisk

Byl vytvořen potomek třídy *AData* s názvem *IFZ*. Třída definuje abstraktní metody z rodičovské třídy. Metoda pro informační zisk přebírá na vstupu jeden argument, a to název predikované hodnoty zvolené uživatelem programu.

V metodě se podle vzorce pro informační zisk spočtou nejprve jednotlivé entropie pro každého prediktora, zaokrouhlené na čtyři desetinná místa, a teprve v následujícím kroku se všechny spočtené hodnoty odečtou od predikovaného atributu daného argumentem na vstupu metody. Výsledek se ukládá s názvem prediktora do slovníku. Ten je výstupem z metody. Algoritmus znázorňuje obrázek 5.3.1.1 Vývojový diagram IFZ.

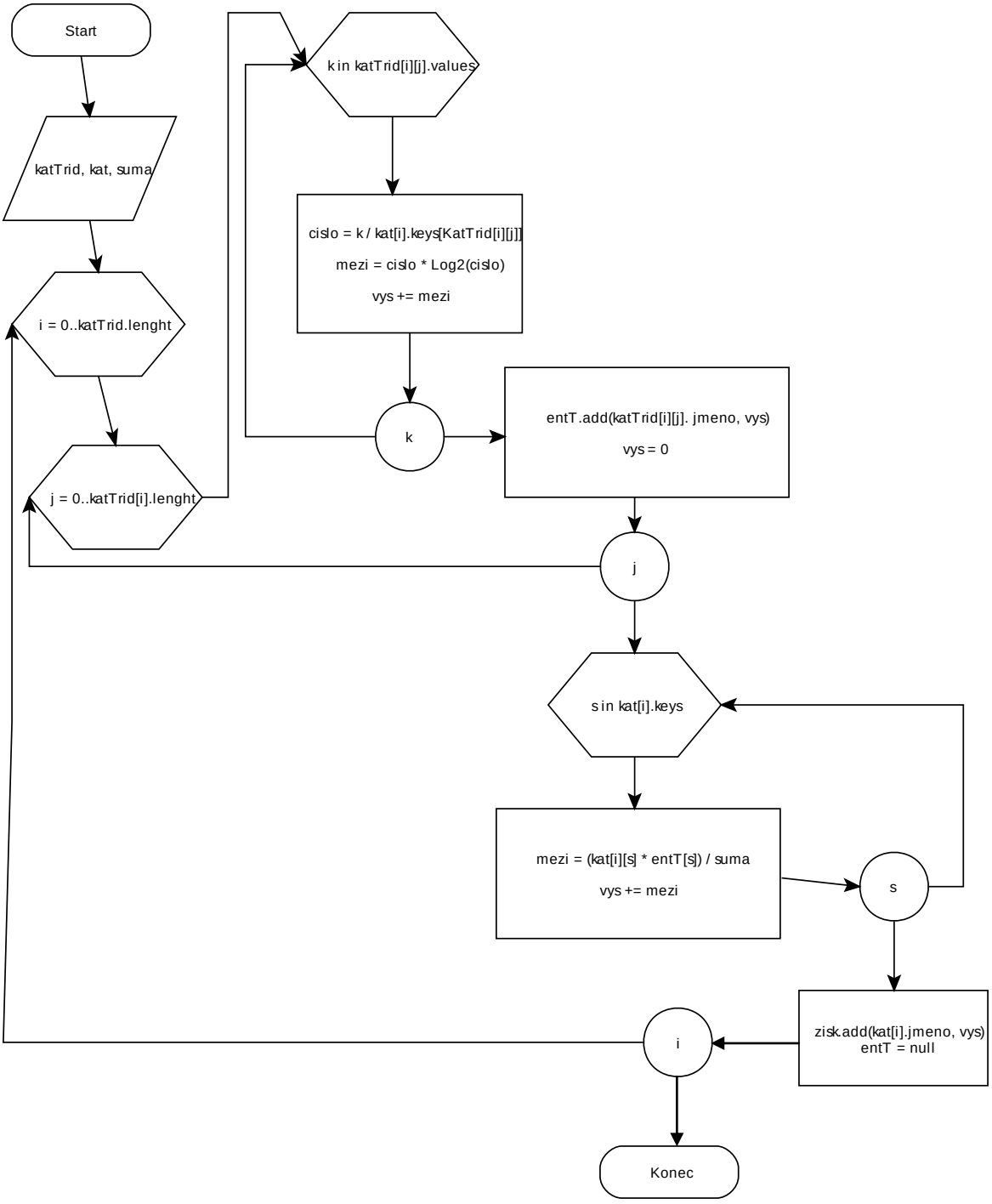


5.3.1.1 Vývojový diagram IFZ

5.3.2 Metoda pro Entropii

Opět byl vytvořen potomek třídy *AData* s názvem *Entropy.* Třída definuje abstraktní metody z rodičovské třídy. Na rozdíl od metody pro informační zisk, ale metoda nepřebírá žádný argument.

V prvním kroku metody se procházejí všechny spočítané četnosti tříd v závislosti na predikovaném atributu, to je popsáno v předchozí kapitole 5.2.2 a počítá pro každou třídu prediktora jeho entropii. Získanou hodnotu ukládá do pomocné proměnné typu slovník. Teprve pak se provede konečný výpočet, kde se znásobí mezi sebou relativní četnost daného atributu s četností tříd dle predikovaného atributu a to se vydělí celkovým počtem záznamů v tabulce, tedy hodnotou *n* dle vzorce. Výsledek je opět zapsán do slovníku, kde klíčem je název prediktora. Algoritmus znázorňuje obrázek 5.3.2.1 Vývojový diagram Entropie.



5.3.2.1 Vývojový diagram Entropie

5.4 Tvorba stromu

S tvorbou rozhodovacího stromu souvisí hned dvě další třídy. Třída Rozpad a třída Uzel.

Třída Uzel reprezentuje jednotlivé listy stromu, tedy to, jak se trénovací data postupně rozpadají do menších tabulek. Obsahuje jen proměnné pro nastavení jména uzlu podle jména prediktora, jména jeho tříd a jeho status, který rozhoduje o tom, zda daný uzel bude použit pro další větvení stromu nebo zůstane listem.

Třída Rozpad je oproti třídě Uzel podstatně složitější. Zde se totiž provádí samotný rozpad původní tabulky na menší podle výsledků spuštěného algoritmu a zároveň se nastavuje seznam jednotlivých uzlů pro vykreslení stromu. Třída obsahuje dvě stěžejní metody. A to metodu *rozpad()* a metodu *getTable()*.

Třída Rozpad je na počátku tvořena proměnnou typu Kategorie, která reprezentuje nejlepšího prediktora, jež byl spočítán po průběhu některého klasifikačního algoritmu, a pak aktuální tabulkou s konkrétními hodnotami.

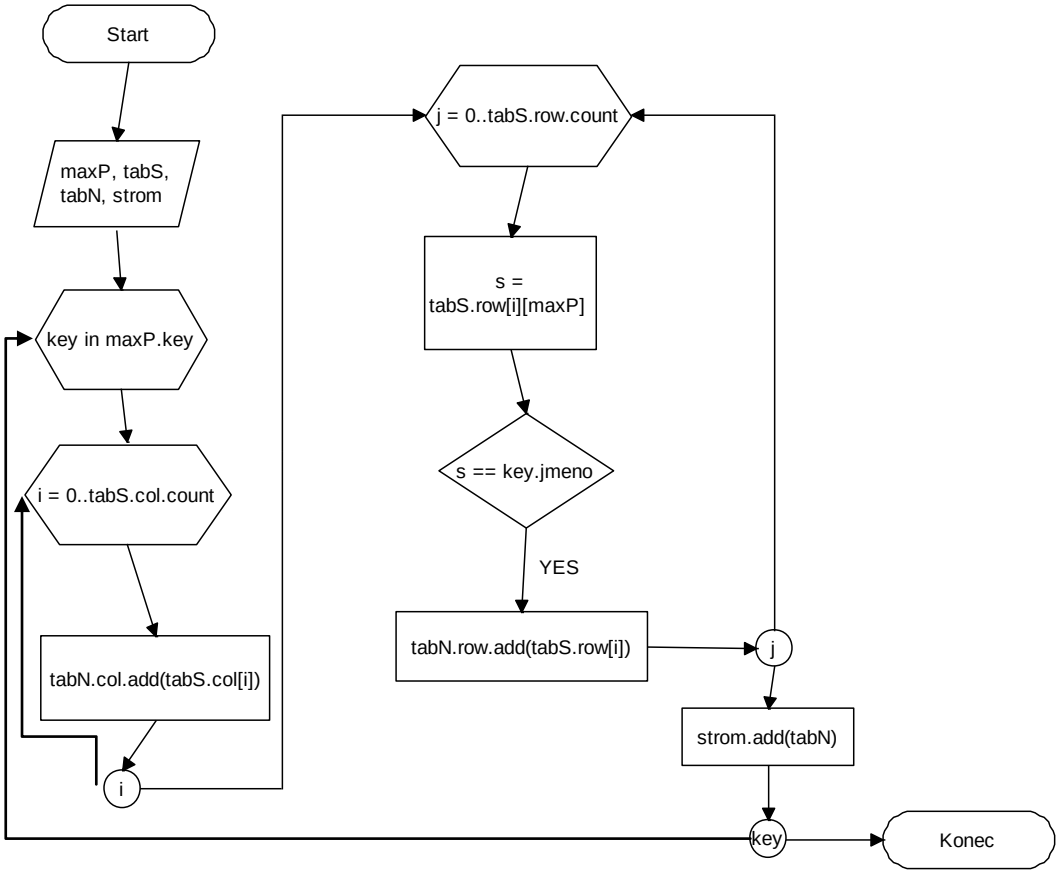
5.4.1 Metoda rozpad()

Metoda *rozpad()* nepřebírá na vstupu žádný argument a nemá návratovou hodnotu. Výsledky metody jsou totiž rovnou zaznamenávány do nového okna programu, které se pak uživateli zobrazí. Výsledné tabulky jsou rozděleny podle jednotlivých tříd nejlepšího prediktora, které jsou zaznamenány v jeho slovníku četností.

Metoda využívá struktury List, což je dynamický seznam hodnot, kam ukládá výsledky, tedy rozpadlé tabulky. Toho je docíleno tak, že se postupně prochází slovník kategorie nejlepšího prediktora. Délka daného slovníku říká, kolik bude ve výsledku tabulek.

Při průchodu tříd metoda rovnou nastavuje Uzel, který na konci přidá do seznamu Listů. Zároveň nastavuje jména sloupců podle původní tabulky a v dalším cyklu postupně prochází všechny řádky tabulky. Zde provede porovnání všech hodnot ve sloupečku nejlepšího prediktora s aktuální hodnotou procházené třídy. Pokud dojde ke shodě, nastaví se do nové

tabulky celý řádek, který je právě procházen. Po skončení cyklu je tabulka přidána do okna pro uživatele a metoda se posouvá k dalšímu páru ve slovníku nejlepšího prediktora. Postup je naznačen na obrázku 5.4.1.1 Vývojový diagram rozpadu.

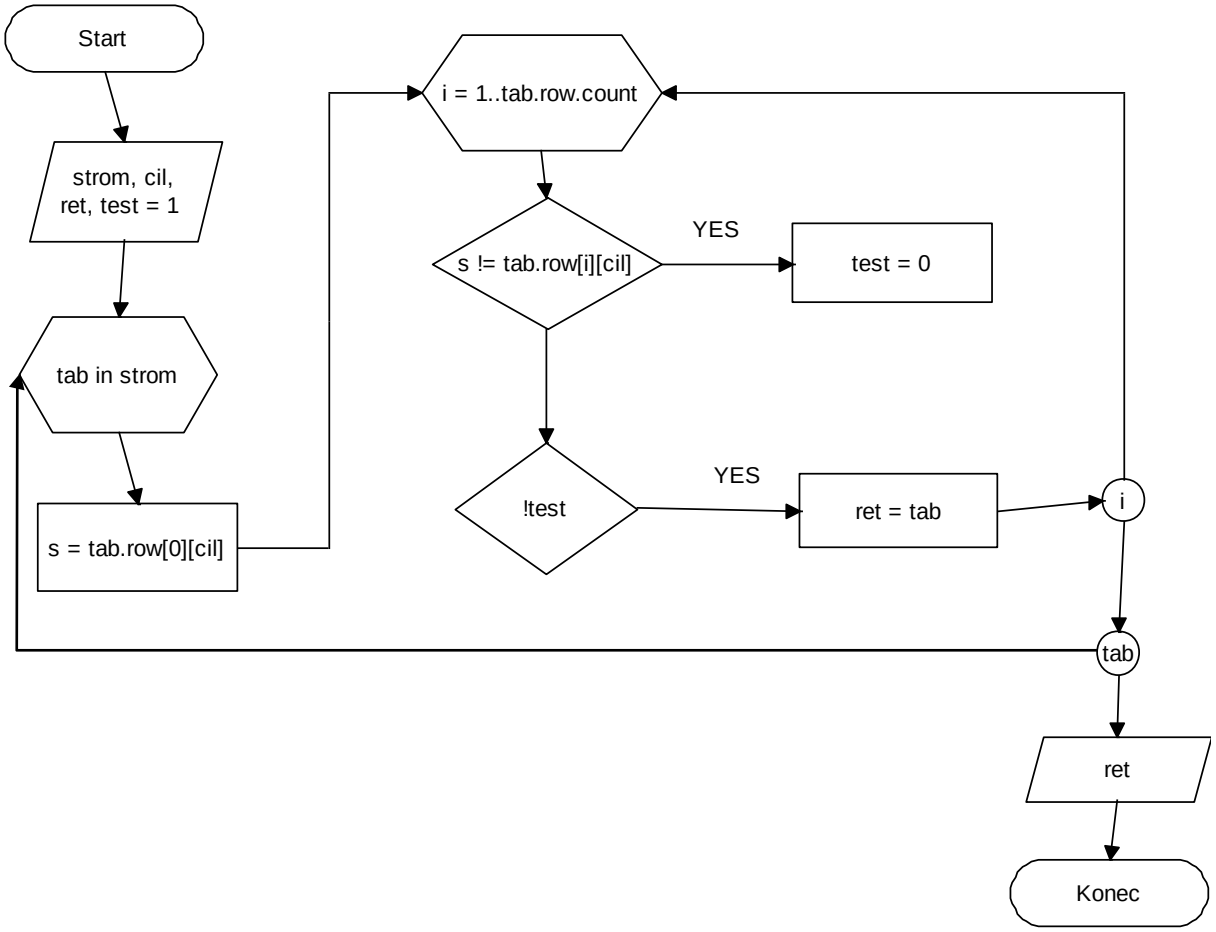


5.4.1.1 Vývojový diagram rozpadu

5.4.2 Metoda getTable()

Tato metoda je důležitá pro další krok klasifikačních algoritmů. Určuje totiž, která z nově vzniklých tabulek po rozpadu má být jako další vstupní tabulkou do daného algoritmu.

Metoda pracuje na principu průchodu řádků predikovaného atributu v každé po rozpadu vzniklé tabulce a kontroluje, zda všechny řádky mají stejnou hodnotu. Pokud nemají, určí tuto tabulku jako další pro vstup a zároveň najde Uzel, který splňuje stejné podmínky a nastaví mu status, který říká, že se od něj bude odvíjet další větvení stromu pro vykreslení. Princip znázorňuje obrázek 5.4.2.1 Vývojový diagram pro vrácení nové tabulky.



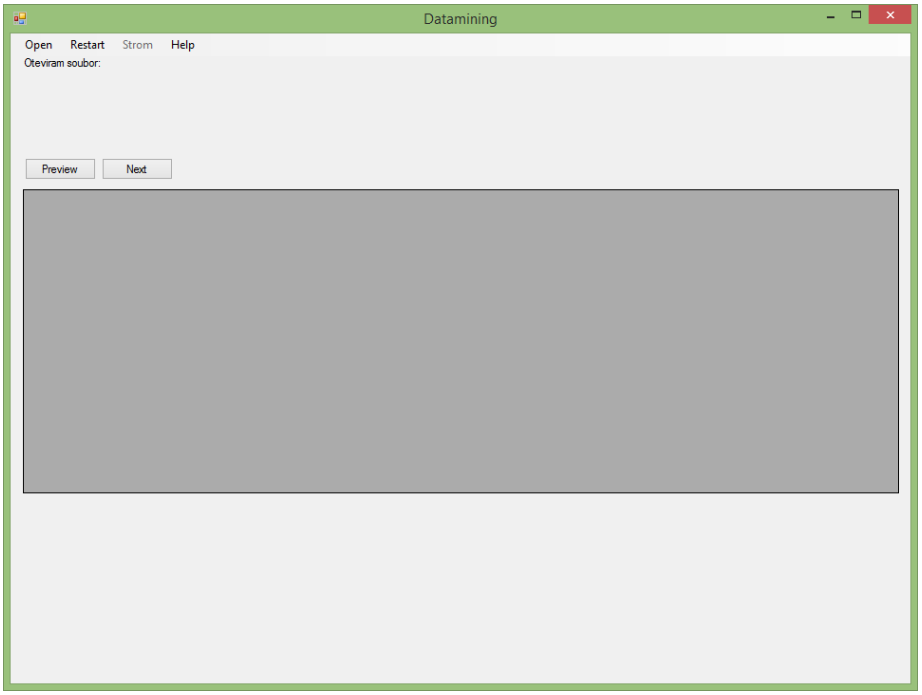
5.4.2.1 Vývojový diagram pro vrácení nové tabulky

5.5 Hlavní část programu

Hlavní část programu propojuje všechny již popsané části a vytváří design aplikace, se kterou uživatel pracuje. Je zde opět několik stěžejních metod, které zajišťují funkčnost algoritmů, z nichž dvě už byly popsány v kapitole Kategorizace vstupních dat 5.2.

5.5.1 Nastavení a spuštění programu

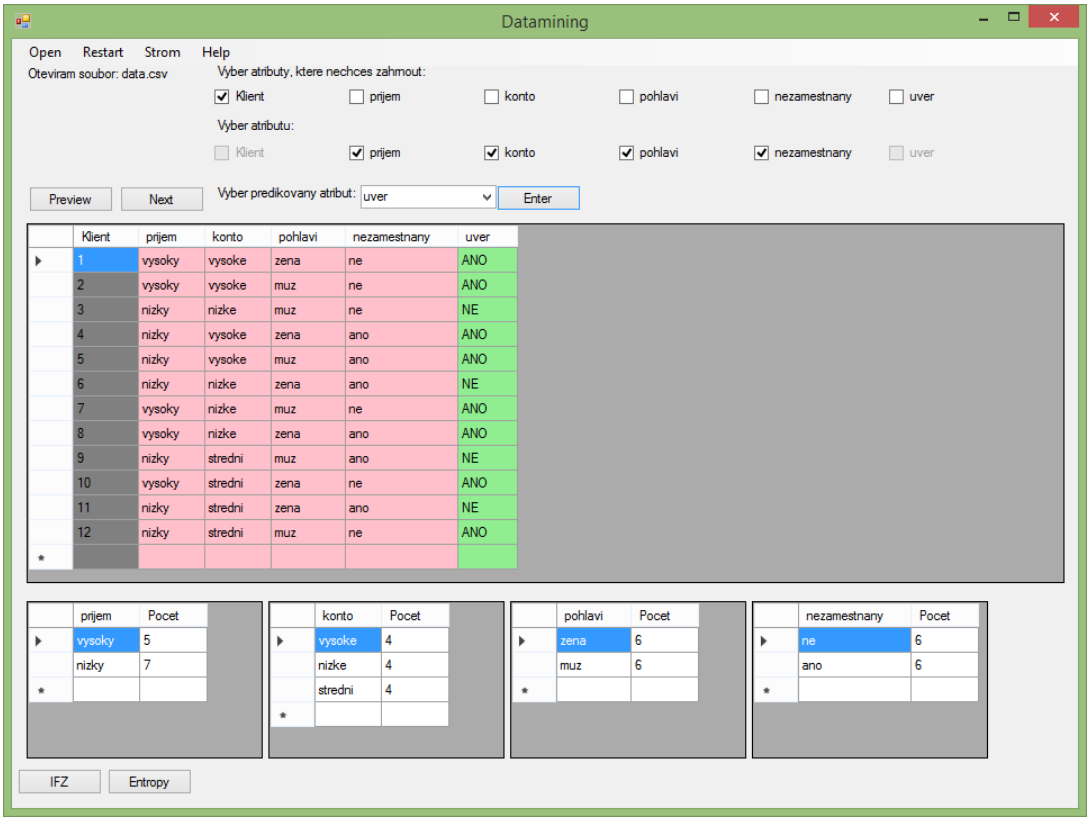
Jak aplikace vypadá po spuštění je vidět na obrázku 5.5.1.1 Po spuštění aplikace. Uživateli je tedy nabízena možnost otevření zdrojového souboru a jeho načtení do tabulky. Aplikace ale pracuje nad testovacími statickými daty. Volba možnosti otevření jiného souboru je zde připravena pro pozdější úpravy.



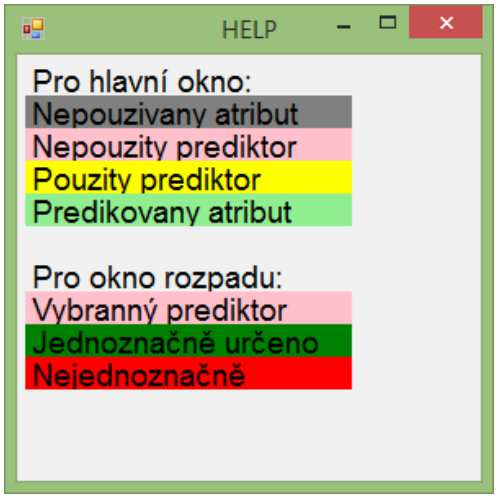
5.5.1.1 Po spuštění aplikace

Po načtení dat dostane uživatel hned několik možností, jak s daty pracovat. Zobrazí se totiž checkboxy se všemi atributy tabulky, kde si uživatel vybírá, které atributy má aplikace vynechat a které naopak použít jako prediktory pro klasifikační algoritmy. Nakonec pak je volba predikovaného atributu. Po potvrzení nastavení se tabulka okamžitě přebarví, zobrazí se relativní četnosti vybraných prediktorů a tlačítka pro spuštění algoritmů Entropie a Informačního zisku, jak je vidět na obrázku 5.5.1.2 Nastavení aplikace pro spuštění algoritmu.

Pro upřesnění barev slouží Help (5.5.1.3 Help), kde jsou všechny barvy vysvětleny.



5.5.1.2 Nastavení aplikace pro spuštění algoritmu



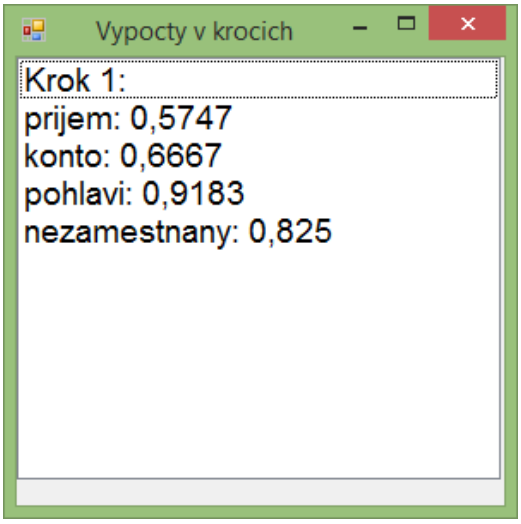
5.5.1.3 Help

5.5.2 Propojení a spuštění algoritmu

Obě metody, které zajišťují spuštění algoritmu Entropie a Informačního zisku, jsou si vcelku podobné. Liší se hlavně v předávání argumentu v metodě pro výpočet. Obě ale využívají konkrétní metodu getmax() pro získání nejlepšího prediktora ze získaných výpočtů.

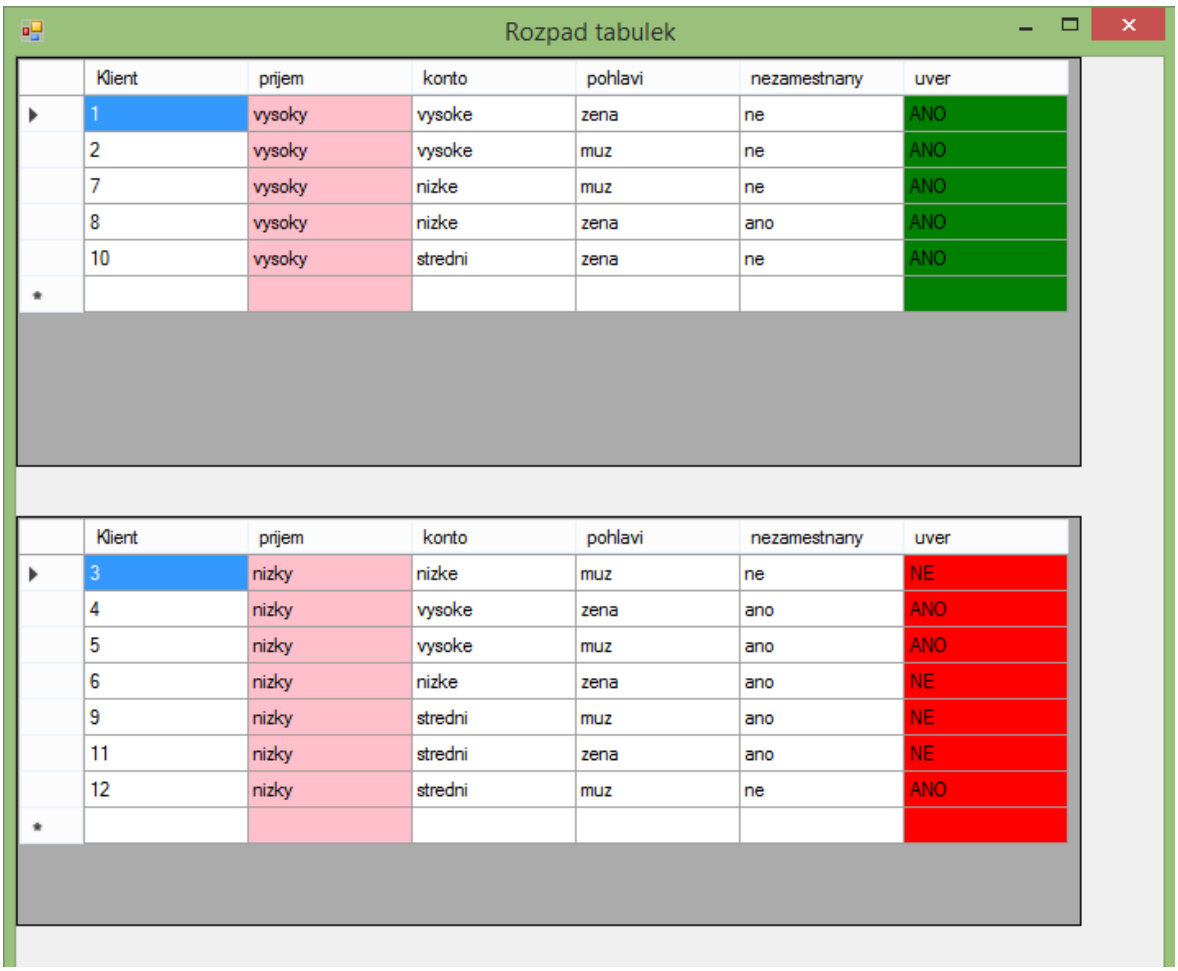
Metoda getMax() pracuje na základě kontroly, který algoritmus byl spuštěn. Podle toho pak v získaných výpočtech vyhledá a vrátí maximální hodnotu pro Informační zisk nebo minimální hodnotu pro Entropii. Zároveň s tím metoda odškrtne v aplikaci vybraného prediktora, aby bylo jasné, že už byl použit.

Samotné propojení po počátečních inicializacích začíná spuštěním daného algoritmu a uložením výsledků získaných z metody výpočet(). Tyto výsledky jsou následně zobrazeny uživateli pomocí nového okna (5.5.2.1 Zobrzaní výpočtů).

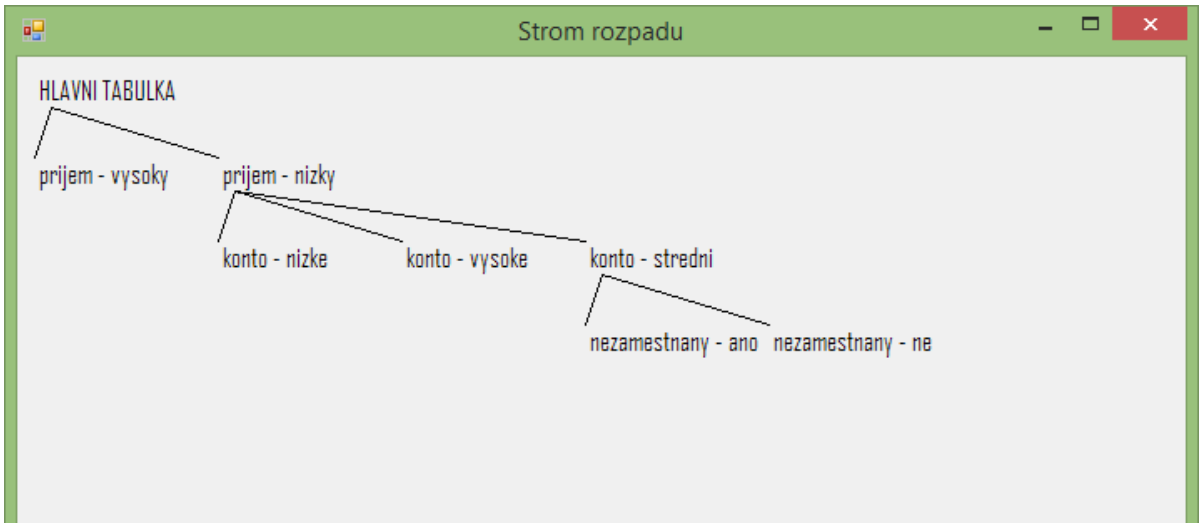


5.5.2.1 Zobrzaní výpočtů

Program pak pokračuje nalezením nejlepšího prediktora a toho posílá spolu s relativními kategoriemi a predikovaným atributem k rozpadu (5.4.1, 5.4.2) a zobrazí v novém okně rozpadlé tabulky (5.5.2.2 Rozpad tabulek). Na základě toho začne vykreslovat rozhodovací strom, který si uživatel může v menu zobrazit (5.5.2.3 Strom rozpadu).



5.5.2.2 Rozpad tabulek



5.5.2.3 Strom rozpadu

Program pokračuje nalezením vhodné tabulky pro pokračování algoritmu. Pro tuto tabulku pak spustí znova celou inicializaci daného algoritmu a čeká na další interakci od uživatele, případně skončí a ohlásí uživateli, že už žádná další tabulka neexistuje.

6 Závěr

Cílem bakalářské práce bylo seznámit se s klasifikačními algoritmy a tvorbou rozhodovacích stromů v dataminigu a vytvořit program pro zobrazení vybraných algoritmů. Pro popis a zobrazení byly vybrány algoritmy Entropie a Informačního zisku.

Oba algoritmy byly nastudovány, naprogramovány a převedeny do vizuální formy. Program je navíc postaven tak, aby mohlo dojít k případnému rozšíření o další algoritmy s využitím stávajících metod. K tomu byla využita šablona Template method založená na abstraktních třídách. Program by mohl být dále rozšířen o algoritmus GINI index s využitím Informačního zisku nebo Entropie. Stejná šablona byla také využita pro případné rozšíření v oblasti otevírání různých typů souborů. Program je také dále připraven pro uživatelské otevírání souborů. Celá aplikace je také spolu s testovacími daty přenositelná.

Algoritmus Entropie pracoval s testovacími daty podle očekávání i podle ručně spočítaných hodnot. Rozděloval tabulky, až zbyly jen takové tabulky, jež splňovaly kritérium pro predikovaný atribut. A to takové, že ve sloupečku predikovaného atributu musí být pouze data stejné třídy.

Na rozdíl od Entropie algoritmus Informačního zisku s daty pracoval jinak. Po prvním kroku algoritmu by se mohlo zdát, že jde vše podle očekávání. Ale při dalším kroku dojde k rozdělení, které kritérium pro predikovaný atribut nesplňuje. Tedy další tabulky nejsou jednoznačně určeny. Přesto jsou výpočty provedeny správně. Byly k tomu provedeny ruční početní kontroly. Je to tedy pravděpodobně dáno testovacími daty, která nemohou být uzpůsobena pro oba algoritmy zároveň.

Z toho lze usoudit, že v reálném dolování z dat nelze používat pouze jen jeden druh algoritmu, nýbrž kombinaci různých klasifikačních algoritmů. S tím ale vzniká problém s předpřípravou dat, protože různé algoritmy vyžadují různé přípravy. V reálných datech je také nutné počítat s problémem chybějících hodnot, který v testovacích datech není nijak uvažován.

Zdroje informací

[1] Young Yin, Ikou Kaku, Jiafu Tang: Data Mining, Springer London Ltd, 2011

[2] Steve McConnell: Dokonalý kód, Computer Press 2006

[3] Kotler Philip: Marketing management. Grada, 2003

[4] Hendl J.: Přehled statistických metod zpracování dat, Portál, 2006

[5] Olivia Parr Rud: Datamining, Computer Press, 2006

[6] Petr Berka: Dobývání znalostí z databází, Academia, 2003

[7] http://www.msps.cz/data-mining/

[8] http://www.spss.cz/pasw_modeler.htm

[9] http://www.algoritmy.net/article/104/Strom

[10] http://www.algoritmy.net/article/1329/Template-method

Příloha obsahu přiloženého CD

Přiložené CD obsahuje kopii bakalářské práce ve formátu PDF s naskenovaným zadáním a podepsaným prohlášením, výslednou aplikaci s testovacími daty ve formátu TXT a CSV a kompletní seznam souborů programu. Celý program je zde pak zazipován pro případné další využití (BP_2015_Franz.zip).

Datová struktura

Text bakalářské práce

text/BP_2015_Franz.pdf

Aplikace

aplikace/ BP_Franz_2015.exe

aplikace/data.txt

aplikace/data.csv

Jednotlivé soubory programu jsou v další složce aplikace/soubory.

AData.cs	- datová struktura pro algoritmy
CSVOpen.cs	- potomek třídy Open
Entropy.cs	- potomek třídy AData
Form1.cs	- hlavní část programu
IFZ.cs	- potomek třídy AData
Kategorie.cs	- datová struktura pro kategorizaci dat
Open.cs	- datová struktura pro otevírání dat
Rozpad.cs	- datová struktura pro rozpad tabulek
TxtOpen.cs	- potomek třídy Open
TypSoubor.cs	- pomocná třída k hlavnímu programu
Uzel.cs	- datová struktura pro tvorbu stromu