VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

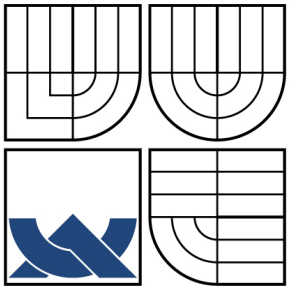
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

PŘÍPRAVA CVIČENÍ PRO DOLOVÁNÍ ZNALOSTÍ Z BÁZE DAT - KLASIFIKACE A PREDIKCE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

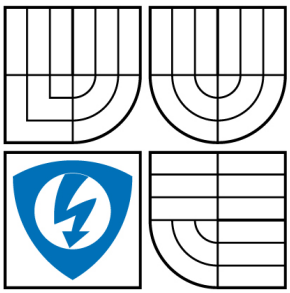
AUTOR PRÁCE Bc. JAN MARTINÍK
AUTHOR

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

PŘÍPRAVA CVIČENÍ PRO DOLOVÁNÍ ZNALOSTÍ Z BÁZE DAT - KLASIFIKACE A PREDIKCE

DESIGN OF EXERCISES FOR DATA MINING - CLASSIFICATION AND PREDICTION

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

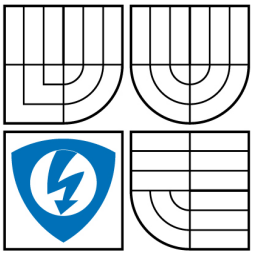
AUTOR PRÁCE — Bc. JAN MARTINÍK

AUTHOR

VEDOUCÍ PRÁCE — Ing. RADIM BURGET

SUPERVISOR

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student:	Bc. Jan Martiník	***ID:***	83496
Ročník:	2	***Akademický rok:***	2008/2009

NÁZEV TÉMATU:

Příprava cvičení pro dolování znalostí z báze dat - klasifikace a predikce

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s technikami pro klasifikaci a predikci dat a jejich možnostmi. Seznamte se s problematikou rozhodovacích stromů. Vytvořte demonstrační cvičení, ve kterém demonstrujte vlastnosti algoritmu. Na školním serveru zprovozněte systém pro dolování informací z báze dat na základě grid technologií.

DOPORUČENÁ LITERATURA:

[1] Data Miners, http://www.data-miners.com/dmt_companion.htm
[2] Ian H. Witten and Eibe Frank, Data Mining: Practical Machine Learning Tools and Techniques, Morgan Kaufmann

Termín zadání:	9.2.2009	***Termín odevzdání:***	26.5.2009

Vedoucí práce: Ing. Radim Burget

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

ABSTRAKT

Diplomová práce na téma „Příprava cvičení pro dolování znalostí z báze dat - klasifikace a predikce“ pojednává o nejčastěji používaných metodách klasifikace a predikce. Mezi metody klasifikace byly zahrnuty asociační pravidla, Bayesovské klasifikace, genetické algoritmy, metoda nejbližšího souseda, neuronové sítě a rozhodovací stromy. Metody predikce obsahují lineární a nelineární regresi. V práci je podrobně shrnuta problematika rozhodovacích stromů a je zde detailně popsán algoritmus pro tvorbu rozhodovacího stromu včetně jednotlivých vývojových diagramů. Navržený algoritmus pro tvorbu rozhodovacího stromu je testován dvěma testy prostřednictvím dat stažených z internetových stránek. Výsledky jsou vzájemně porovnány a jsou popsány rozdíly mezi oběma implementacemi. Práce je napsaná tak, aby čtenář po jejím přečtení získal představu o jednotlivých metodách a postupech při dolování znalostí z báze dat, jejich výhodách, nevýhodách a problematice, která je s nimi úzce spjatá.

KLÍČOVÁ SLOVA

Dolování znalostí, Metody dolování znalostí, Klasifikace, Predikce, Rozhodovací strom

ABSTRACT

My master's thesis on the topic of „Design of exercises for data mining - Classification and prediction“ deals with the most frequently used methods classification and prediction. There are association rules, Bayesian classification, genetic algorithms, the nearest method neighbor, neural networks and decision trees on the classification. There are linear and non-linear prediction on the prediction. This work also contains a summary of detail the issue of decision trees and a detailed algorithm for creating the decision tree, including development of individual diagrams. The proposed algorithm for creating the decision tree is tested through two tests of data downloaded from Internet. The results are mutually compared and described differences between the two implementations. The work is written in a way that would provide the reader with a notion of the individual methods and techniques for data mining, their advantages, disadvantages and some of the issues that directly relate to this topic.

KEYWORDS

Data mining, Data mining methods, Classification, Prediction, Decision tree

MARTINÍK, J. *Příprava cvičení pro dolování znalostí z báze dat - klasifikace a predikce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií. Ústav telekomunikací, 2009. 65 s. Vedoucí diplomové práce Ing. Radim Burget.*

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Příprava cvičení pro dolování znalostí z báze dat - klasifikace a predikce“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne ....

(podpis autora)

PODĚKOVÁNÍ

Děkuji vedoucímu diplomové práce Ing. Radimovi Burgetovi za velmi užitečnou metodickou pomoc a cenné rady při zpracování diplomové práce.

V Brně dne ....

(podpis autora)

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

V dneštní době se ve všech odvětvích produkuje stále větší množství dat a jednotlivé organizace si začínají uvědomovat, že jejich zpracování spolu s hlubší analýzou může vést k dalšímu účelnému mnohdy i cílenému využití.

Průkopníky dolování znalostí z báze dat byly v minulosti hlavně univerzity a různé vědecké kruhy, ale postupem času, kdy jeho vliv začínal stále více sílit a slavit úspěch, se toto téma začalo dostávat i do ziskového sektoru. Nejčastěji se dnes využívá dolování znalostí z báze dat v marketingových strategiích, výrobních programech, kapitálových investicích a obecně všude tam, kde pomáhá v zostřeném konkurenčním boji a tím udržuje schopnost konkurence a adaptace daného subjektu. Správná analýza dat totiž odkrývá informační proces vedoucí k získání velmi cenných informací, které lze v konečné fázi efektivně využít k navýšení zisků.

Mezi obory, ve kterých se dolování informací z báze dat stále více uplatňuje patří pojišťovnictví, bankovnictví, telekomunikace, energetika, veřejné služby, zásilkové služby, maloobchod, farmaceutický průmysl, vědní disciplíny a mnohé další.

Dolování znalostí z báze dat je ze základu postaveno na dvou formách datové analýzy - klasifikaci a predikci.

V obecném měřítku využívá dolování znalostí z báze dat propojení poznatků ze statistiky, databázových systémů a strojového učení. Průnik těchto tří oblastí není zdaleka snadný a jednoduchý, ale protože v budoucnu bude stále více růst objem produkovaných dat, bude dolování znalostí z báze dat, i za těchto podmínek, nabírat stále většího významu.

1 KLASIFIKACE A PREDIKCE

Získávání informací z báze dat je opakující se proces zkoumání a analýzy velkého množství dat (např. analýza databáze), při kterém jde o to, aby byly v datech objeveny na první pohled skryté informace, vedoucí ke stanovení smysluplných vzorů a pravidel, které následně vedou k aplikaci inteligentních rozhodnutí. Více o této problematice se lze dočíst v [20], [12]. Celá metodika by měla zahrnovat pokud možno co nejlepší instrukce, které by měly předcházet učení věcí, které nejsou pravdivé, nebo učení věcí, které jsou pravdivé, ale nejsou užitečné.

Klasifikace a predikce jsou dvě formy z datové analýzy, které mohou být použity pro získání popisujícího modelu důležitých datových tříd nebo predikce budoucích datových trendů. Taková analýza může pomoci lépe porozumět rozsáhlým souborům dat. Zatímco klasifikace předpovídá a určuje kategorie, druhy a třídy, predikce vytváří spojité hodnoty funkcí. Mnoho klasifikačních a predikčních metod je založeno na poznatcích z oblastí strojového učení, statistiky a hledání vzorů.

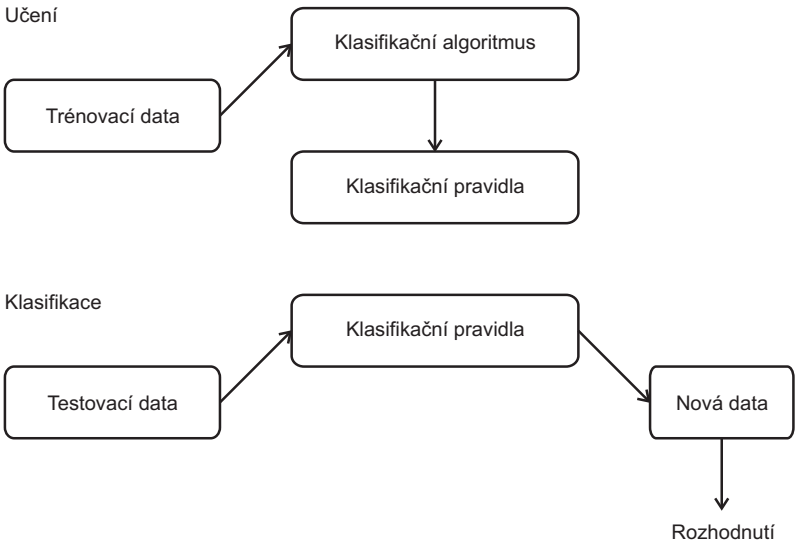
1.1 Klasifikační proces

Části klasifikačního procesu:

- učení - trénovací data jsou analyzována klasifikačním algoritmem, naučený model je reprezentován klasifikačními pravidly;
- klasifikace – u testovacích dat je přesnost ohodnocována klasifikačními pravidly a jestliže je přijatelná, mohou být pravidla aplikována ke kvalifikaci nových dat.

První krok klasifikačního procesu představuje učení se charakteristiky nebo funkce $y = f(X)$, která představuje atribut asociační třídy y závislé na X. Jde tedy o separaci dat do příslušných tříd. Tuto funkci zastávají klasifikační pravidla, rozhodovací stromy nebo matematické rovnice. V druhém kroku je model použitý pro klasifikaci.

Zjednodušeně se dá říci, že ve fázi učení jde o to, aby se z trénovacích dat vytvořily za pomocí klasifikačního algoritmu klasifikační pravidla. Ve fázi klasifikace se s využitím testovacích dat a klasifikačních pravidel získá rozhodnutí o příslušných nových datech. Klasifikační proces je zobrazen níže (viz. obr. 1.1).



Obrázek 1.1: Klasifikační proces: učení, klasifikace

1.2 Příprava dat pro klasifikaci a predikci

Pro lepší přesnost, výkonnost a rozšířitelnost klasifikačního nebo predikčního procesu se používá následujících předzpracování dat:

- očištění dat - jde o předzpracování dat úpravou odstranění nebo redukování šumu a ošetření chybějících hodnot (nahrazení chybějících hodnot nejčastějšími hodnotami atributů nebo nejvíce pravděpodobnými statistickými hodnotami); zatímco mnoho klasifikačních algoritmů má nějaké mechanismy pro zacházení s šumem nebo chybějícími daty, tento krok má pomoci redukovat nepořádek během učení;

- relevanční analýza - mnoho atributů v datech může být nadbytečných; korelační analýza vychází z toho, jestli nějaké dva dané atributy jsou statisticky příbuzné;

- transformace a redukce dat - data mohou být transformována pomocí normalizace, zvláště když jsou k jednotlivým učebním krokům použity neuronové sítě nebo metody zahrnující vzdálenostní měření; normalizace zahrnuje normování všech hodnot pro dané atributy, které spadají do malého specifického rozsahu.

Data mohou být také redukována použitím mnoha jiných metod vycházejících např. z waveletové transformace nebo principů analýzy komponent diskretizačních technik, jako jsou analýza histogramu a shlukování.

2 METODY KLASIFIKACE A PREDIKCE

Mezi základní techniky klasifikace dat patří asociační pravidla, Bayesovské klasifikace, genetické algoritmy, metoda nejbližšího souseda, neuronové sítě a rozhodovací stromy. Metody predikce zahrnují lineární regresi, nelineární regresi a další méně významné regresně orientované metody. Konkrétní popisy jednotlivých metod lze nalézt v [6] nebo [8].

V jednotlivých aplikacích klasifikace a predikce u dolování informací z báze dat neexistuje žádný jednoduchý přístup, ale většinou se využívá sad metod, které se používají samostatně nebo se navzájem kombinují.

2.1 Vlastnosti klasifikačních a predikčních metod

Klasifikační a predikční metody mohou být porovnávány a ohodnocovány podle následujících kritérií:

- přesnost - přesnost klasifikátoru představuje schopnost daného klasifikátoru správně určovat třídy; přesnost prediktoru odkazuje na to, jak dobře daný prediktor odhaduje hodnoty předpovězených atributů; vlastnosti a početní operace týkající se přesnosti lze nalézt v [8];
- rychlost - označuje výpočetní nároky při generování a používání daných klasifikátorů a prediktorů;
- odolnost - je schopnost klasifikátoru nebo prediktoru vytvářet správné predikce i přes výskyt šumových dat nebo chybějících hodnot;
- rozšířitelnost - vlastnost klasifikátoru nebo prediktoru zacházet efektivně s velkým množstvím dat;
- vyložitelnost - schopnost porozumění a celkového pohledu poskytnutých klasifikátorem nebo prediktorem; vyložitelnost je hodně subjektivní.

2.2 Příklady použití metod klasifikace a predikce

Metody klasifikace a predikce se v dneštní době při dolování informací z báze dat nejčastěji vyskytují v ekonomice (predikce a analýza úvěrového rizika, vydávání kreditních karet), bankovnictví (bankovní převody – detekce podezřelých finančních transakcí), telekomunikaci (monitoring a analýza informací o jednotlivých hovorech, geografické poloze), službách (supermarkety a velkoprodejny – jaká zboží

jsou prodávána současně a s jakým výskytem), kriminalistice (záznamy zločinců), lékařství apod.

2.3 Asociační pravidla

Asociační pravidla jsou techniky dolování informací z báze dat, které jsou nejvíce spojené s tzv. analýzou nákupního košíku. Při analýze nákupního košíku se zjišťuje, které položky si zákazníci kupují současně, což v oblasti datových sad záznamů odpovídá hledání vzájemných vazeb (asociací) mezi jednotlivými daty. Jde o pravidla, která se vyskytují ve všech programovacích jazycích (konstrukce IF-THEN) a také v běžné mluvě. Společně s rozhodovacími stromy patří k nejčastěji používaným prostředkům k reprezentaci znalostí, protože oba tyto prostředky jsou oproti ostatním hodně intuitivní. Více o asociačních pravidlech lze nalézt v [6].

2.3.1 Druhy asociačních pravidel

U asociačních pravidel existují tyto pravidla:

- vypovídající pravidlo – obsahuje vysoce kvalitní a vypovídající informace;
- banální pravidlo – informace, která je dobře známá lidem obeznámeným s danou problematikou;
- nevysvětlitelné pravidlo – neexistuje vysvětlení, pro jeho pochopení se podniká určitá návrhová akce.

Banální a nevysvětlitelná pravidla se vyskytují nejčastěji.

2.3.2 Charakteristiky a vlastnosti asociačních pravidel

Vlastnosti asociačních pravidel:

- spolehlivost (důvěra) – procento počtu všech objektů ku počtu objektů vyhovujícím dané podmínce; jinak řečeno jde o podmíněnou pravděpodobnost závěru, pokud platí předpoklad; pokud je předpoklad platný, je podmíněn;
- podpora – počet objektů, které splňují předpoklad i závěr.

Asociační pravidla mají za úkol hledat všechny možné asociace (ekvivalence apod.) mezi hodnotami jednotlivých atributů, což je docíleno pomocí asociačních algoritmů. Tyto algoritmy generují příslušné kombinace (popř. konjunkce). Při generování prohledáváme prostor všech přípustných konjunkcí do šířky nebo do hloubky.

Prohledávání do šířky pracuje tak, že se generují nejprve všechny kombinace délky jedna, potom kombinace délky dvě, atd.

Prohledávání do hloubky je založeno na vygenerování kombinace délky jedna a ta se prodlužuje (o první kategorii dalšího atributu). Pokud to již nelze provést, znamená to, že se vyčerpaly kategorie posledního atributu, kombinace se zkrátí a změní se poslední kategorie.

2.3.3 Automatické generování asociačních pravidel

Asociační pravidla mohou být automaticky generována pokud:

- představují vzory v datech bez specifikované cílové proměnné;
- jde o neřízené (automatické) dolování informací;
- je roboticky rozhodnuto, že vzory dávají smysl.

Často se opakující položky a jejich odpovídající asociace nebo korelační pravidla charakterizují zajímavé vztahy mezi atributy a třídami. Z tohoto důvodu mohou být použity pro efektivní klasifikaci. Asociační pravidla ukazují silné asociace mezi hodnotami atributů, které se často vyskytují v sadách dat. Nejčastěji používané algoritmy u asociačních pravidel jsou CBA a CMAR.

Algoritmus CBA

Tento algoritmus patří mezi nejjednodušší algoritmy asociační klasifikace. Data jsou vícenásobně procházeny, odvozené časté množiny položek jsou používány ke generování a testování delších množin položek. Čísla průchodů se rovnají délce nejdelších nalezených pravidel. Algoritmus CBA [10] používá heuristických metod ke konstruování klasifikátoru, kdy jsou pravidla organizována podle klesající priority na důvěryhodná a únosná. Pokud má sada pravidel nějakého předchůdce, je pravidlo s nejvyšší důvěryhodností vybráno k reprezentování této sady.

Algoritmus CMAR

Algoritmus CMAR [15] od algoritmu CBA odlišuje strategie dolování četnosti množiny položek a konstrukce klasifikátoru. Používá stromovou strukturu k efektivnímu získání a uložení pravidel a navíc i prořezávací pravidla.

2.3.4 Výhody a nevýhody

Výhody asociačních pravidel:

- v mnoha případech nacházejí lepší výsledky než tradiční klasifikační metody (např. C4.5 u rozhodovacích stromů).

2.4 Bayesovské klasifikace

Bayesovské klasifikace patří mezi statistické klasifikace [13]. Mají za úkol předpovídat pravděpodobnosti členů tříd. Při mnoha studiích byl nalezen jednoduchý Bayesův klasifikátor nazývaný prostý Bayesův klasifikátor, který je výkonově srovnatelný s rozhodovacími stromy a s vybranými klasifikátory neuronových sítí. Bayesovy klasifikátory také vykazují vysokou přesnost a rychlost při použití ve velkých databázích. Bayesovské klasifikace jsou založeny na Bayesově větě [21].

2.4.1 Bayesova věta

$$P(H \mid X) = \frac{P(X \mid H) \times P(H)}{P(X)}$$

Kde H je hypotéza a X je evidence, $P(H)$ a $P(X)$ jsou jejich pravděpodobnosti.

2.4.2 Prosté Bayesovské klasifikace

Prosté Bayesovy klasifikátory předpokládají, že efekt, který mají hodnoty jednoho atributu na dané třídě jsou nezávislé na hodnotách ostatních atributů. Jinak řečeno jde o podmíněnou pravděpodobnost, která vychází z definice Bayesovy věty a kterou lze zapsat vztahem:

$$P(C_i \mid X) = \frac{P(X \mid C_i) \times P(C_i)}{P(X)}$$

Kde C_i představuje různé třídy.

2.4.3 Bayesovské sítě

Bayesovské sítě specifikují propojení distribucí podmíněných pravděpodobností. Poskytují grafický model kauzálních vztahů, na kterém může být vykonáno učení. Trénované Bayesovské sítě mohou být použity ke klasifikaci. Bayesovské sítě jsou také známy pod pojmem pravděpodobnostní sítě.

U Bayesovských sítí jsou definovány dvě komponenty - řízený neperiodický diagram a tabulka podmíněných pravděpodobností. Každý uzel je řízený neperiodickým

diagramem reprezentujícím náhodnou proměnnou. Proměnné musí být diskrétní nebo spojité a odpovídají aktuálním atributům daným v datech nebo skrytých proměnných. Každá proměnná je podmíněně pravděpodobná bez potomků v diagramu, má pouze rodiče.

Uzel uvnitř sítě může být vybrán jako výstupní uzel reprezentující atribut třídy. Lze vybrat více než jeden výstupní uzel. Na síť mohou být aplikovány různé algoritmy učení. Při návratu jména jedné třídy vrací klasifikační proces pravděpodobnostní rozdělení dané pravěpodobností každé třídy.

Trénování Bayesovských sítí

Při trénování Bayesovské sítě se počítá, který ze scénářů je vhodný. Topologie sítě (vrstvy uzlů) musí být dána předem nebo musí být odvozená z dat. Proměnné sítě musí být všechny viditelné nebo skryté. Souhrn skrytých dat by zahrnoval chybějící hodnoty nebo nekompletní data.

Některé existující algoritmy pro učení topologie sítě z trénovacích dat dávají viditelné proměnné, problém je ale většinou s diskrétní optimalizací. Když je známa topologie sítě a viditelné proměnné, tak je trénovací síť zřetelná. Počítání probíhá na položkách tabulky podmíněných pravděpodobností, podobně jako počítání pravděpodobností v prosté Bayesovské klasifikaci. Když je ale dána topologie sítě a některé proměnné jsou skryté, vybírá se z různých metod pro trénování sítě.

2.4.4 Výhody a nevýhody

Výhody Bayesovských klasifikací:

- mají minimální výskyt chyb v porovnání s ostatními klasifikátory;
- poskytují teoretické ospravedlnění ostatním klasifikátorům, které nejsou výlučně Bayesovského typu (založeny na Bayesově větě).

Nevýhody Bayesovských klasifikací:

- při použití mohou vznikat nepřesnosti v předpokladech;
- někdy dochází k nedostatku dostupnosti pravděpodobných dat.

2.5 Genetické algoritmy

Genetické algoritmy jsou heuristické postupy patřící mezi metody strojového učení, které stejně jako například neuronové sítě nevyužívá statistického rozdělení. Jsou

to algoritmy založeny na evolučním procesu přežití nejpřizpůsobivějšího – určitý druh se během svého vývoje zdokonaluje tím způsobem, že se z generace na generaci přenáší genetická informace jen nejsilnějších jedinců a to tak, že informace může přecházet přímo, s určitou modifikací nebo prostřednictvím potomků.

Počáteční populace je vytvářena skládáním náhodně generovaných pravidel. Každé pravidlo může být reprezentováno řetězci bitů.

2.5.1 Principy evolučních procesů

Existují následující principy evolučních procesů:

- dědičnost – je proces, kdy se z rodičovského objektu získávají vlastnosti nebo predispozice pro potomka;
- mutace - je dědičná změna genetické informace vzniklá chybou vyvolanou určitým mutagenem;
- přirozený výběr - je proces výběru jedince z různorodé skupiny jedinců pomocí určitého kritéria, které daného jedince buď potlačuje nebo zvýhodňuje;
- křížení - je proces rozmnožování určitého druhu, kdy se vychází z různých vlastností a predispozicí.

Kritériem hodnocení jedinců v populaci je kvalita toho daného řešení, která se vyjadřuje pomocí fitness funkce. Přepis fitness funkce může být neměnný po celou dobu evoluce, nebo se naopak může měnit pro každou novou generaci. U dolování informací jde o přesnost hypotézy při kvalifikaci. Více informací lze nalézt v [7].

2.5.2 Výhody a nevýhody

Výhody genetických algoritmů:

- jsou ideální pro použití v kombinaci s ostatními technikami dolování dat.

Nevýhody genetických algoritmů:

- vyžadují výkonově náročné počítačové zpracování.

2.6 Lineární regrese

Lineární regrese provádí aproximaci hodnot přímkou (polynomem 1. řádu) metodou nejmenších čtverců.

Přímočará regresní analýza zahrnuje odpovědi proměnné y a jednotlivé proměnné prediktoru x. Je to nejjednodušší forma regrese a modely y představují lineární závislost x. Platí vzorec:

$$y = b + wx$$

Kde se předpokládá, že y může být konstanta, b a w jsou regresní koeficienty, které mohou být reprezentovány váhami. Vztah pak lze přepsat na:

$$y = w_0 + w_1 x$$

Tyto koeficienty se řeší pro metodu nejmenších čtverců, která umožňuje nalézt vhodnou aproximační funkci pro dané hodnoty. Když bude D trénovací sada skládající se z hodnot proměnných prediktoru x obsahovat referenční body formy $(x_1, y_1), (x_2, y_2), \ldots, (x_{|D|}, y_{|D|})$, mohou být regresní koeficienty odhadnuty použitím:

$$w_1 = \frac{\sum_{|D|}^{i=1}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{|D|}^{i=1}(x_i - \bar{x})^2}$$

Kde $\bar{x}$ označuje hodnoty $x_1, x_2, \ldots, x_D$ a $\bar{y}$ hodnoty $y_1, y_2, \ldots, y_D$. Koeficienty w_0 a w_1 často poskytují aproximaci komplikovaných regresních rovnic.

2.6.1 Vícenásobná lineární regrese

Vícenásobná lineární regrese představuje rozšíření lineární regrese tak, že zahrnuje více než jednu proměnnou prediktoru. To dovoluje odpovědím proměnné y být formovány prostřednictvím lineární funkce. Proměnné nebo atributy n prediktorů $A_1, A_2, \ldots, A_n$ lze popsat jako $X = (x_1, x_2, \ldots, x_n)$. Trénovací sada dat D obsahuje data ve tvaru $(X_1, y_1), (X_2, y_2), \ldots, (X_{|D|}, y_{|D|})$. Vícenásobnou lineární regresi se dvěmi proměnnými nebo atributy A_1 a A_2 jde napsat vztahem:

$$y = w_0 + w_1 x_1 + w_2 x_2$$

Kde x_1 a x_2 jsou hodnoty atributů A_1 a A_2. Metoda nejmenších čtverců ukazuje jak až může být rozšířeno řešení.

2.7 Metoda nejbližšího souseda

Metoda nejbližšího souseda patří mezi neparametrické metody klasifikace, u nichž nepředpokládáme znalost tvaru pravděpodobnostní charakteristiky. U metody nejbližšího souseda jsou třídy reprezentovány svými typickými představiteli. Tato metoda vychází ze shlukové analýzy.

2.7.1 Definice metody nejbližšího souseda

Neznámý prvek a zařadíme do stejné třídy jako a_R pokud platí:

$$\lfloor a_R - a \rfloor = min \lfloor a_i - a \rfloor_{i=1..n}$$

kde $(a_i, c_i)_{i=1..n}$ je trénovací množina, n je velikost této trénovací množiny a a_i je vzorek, kterému je přiřazena třída c_i.

Algoritmus nejbližšího souseda je hodně jednoduchý, prakticky jde o porovnání prvků trénovací a testovací množiny. V procesu klasifikace je tedy nový reprezentant zařazen do třídy na základě podobnosti, kterou udává nejmenší vzdálenost tohoto reprezentanta a dané třídy. Paměťová náročnost je úměrná celkovému počtu prvků v množině a její zmenšení lze provést vhodnou minimalizací trénovací množiny.

2.7.2 Výhody a nevýhody

Výhody metody nejbližšího souseda:

- není třeba znát rozložení pravděpodobnosti testovaných dat;
- metoda je chybově srovnatelná se složitějšími metodami;
- jednoduchý návrh a implementace.

Nevýhody metody nejbližšího souseda:

- pomalé rozhodování (závisí na použitém algoritmu);
- vysoká paměťová náročnost (pokud se neminimalizuje trénovací množina).

2.8 Nelineární regrese

Nelineární regrese se někdy označuje jako polynomiální regrese. Často se jí využívá při tom, když je třeba právě jedna proměnná prediktoru. Může se modelovat přidáním polynomiální podmínky do jednoduchého lineárního modelu. Použitím transformací na proměnných lze měnit nelineární model na lineární, který se dá lehce vypočítat pomocí metody nejmenších čtverců.

Transformace polynomiálního regresního modelu na lineární regresní model předpokládá kubický polynomiální vztah daný:

$$y = w_0 + w_1 x + w_2 x^2 + w_3 x^3$$

Pro převedení tohoto vztahu na lineární formu můžeme definovat nové proměnné:

$$x_1 = x$$

$$x_2 = x^2$$

$$x_3 = x^3$$

Poté může být vztah převeden na lineární formu a zapsán ve tvaru:

$$y = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3$$

Výsledný vztah je jednoduše řešený metodou nejmenších čtverců za pomocí softwaru určeného pro regresní analýzu. Polynomiální regrese se dá shrnout jako zvláštní případ vícenásobné regrese. Výrazy vyššího řádu x^2, x^3 a tak dále, jednoduchých funkcí jednoduché proměnné x, mohou být považovány jako ekvivalenty k přidávání nových nezávislých proměnných. Pořád ale existují některé nelineární modely, které nemohou být jednoduše převedeny na lineární model.

2.9 Neuronové sítě

Umělé neuronové sítě [5] jsou paralelní distribuované systémy výkonných prvků modelující biologické neurony, které jsou účelně uspořádány tak, aby byl systém schopen požadovaného zpracování informací. Nevychází z žádného statistického rozdělení.

Zjednodušeně jde o neuronových sítích říci, že je to třída silných a univerzálních nástrojů sloužících k předpovědi, třídění a shlukování.

Umělé neuronové sítě vycházejí z nejdokonalejší biologické neuronové sítě – lidského mozku (má schopnost zevšeobecnit zkušenost). Podobně jako mozek je tvoří množství navzájem propojených elementů, tzv. neuronů. V umělých neuronových sítích chápeme neuron jako buňku, která přijímá zvenčí od jiných neuronů popřípadě vnějšího vstupu sítě podněty a pokud tyto podněty překročí určitý práh, neuron se aktivuje a sám začne prostřednictvím svého výstupu působit na ostatní neurony. Tento proces je podobný rozpoznávání vzorů a minimalizaci chyby.

Neuronové sítě byly kritizovány za špatnou interpretaci. Tyto vlastnosti inicializovaly vytvoření menších neuronových sítí, které již byly vhodnější k používání v dolování informací z bází dat. Mohou být také použity když je k dispozici malá znalost vztahů mezi atributy a třídami. Algoritmy neuronových sítí jsou paralelní. Paralelní techniky zrychlují výpočetní proces. V součastnosti existují techniky pro extrakci pravidel z trénovaných neuronových sítí. Tyto faktory přispívají k většímu zužitkování neuronových sítí při klasifikaci a predikci v dolování informací z báze dat.

2.9.1 Charakteristika neuronové sítě

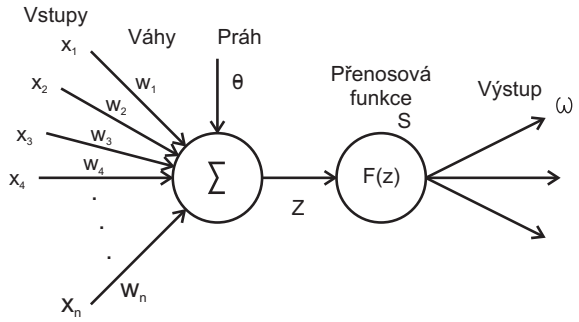
Neuronovou síť můžeme charakterizovat těmito prvky:

- použitý model neuronu;
- topologie neuronové sítě;
- způsob trénování (učení) neuronové sítě;
- způsob vybavování.

Životní cyklus neuronové sítě se skládá z organizační etapy (model neuronu, topologie sítě), adaptivní etapy (učení, trénování) a aktivační etapy (vybavování, odezva).

2.9.2 Model neuronu

Jednotka neuronové sítě představuje model biologického neuronu (viz. obr. 2.1).



Obrázek 2.1: Schéma neuronu

Platí zde vztah:

$$\omega = S(\sum_{i=1}^{n} w_i x_i + \theta)$$

Kde ω je výstup neuronu, θ je práh neuronu, $x = [x_1, x_2, \ldots, x_n]^T$ jsou vstupy neuronu, $w = [w_1, w_2, \ldots, w_n]^T$ jsou váhy neuronu a S je přenosová funkce neuronu definovaná:

$$S(t) = 0; t \in (-\infty, 0)$$

$$S(t) = 1; t \in \langle 0, \infty)$$

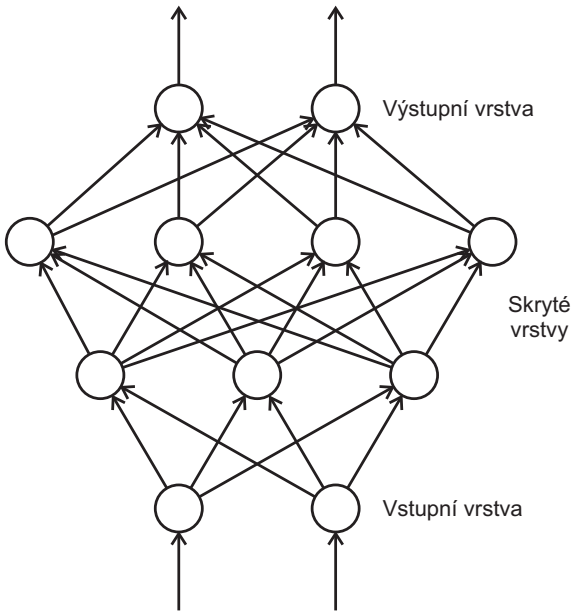
Podrobnější popis neuronových sítí je k dispozici v [14], [2] a [11].

2.9.3 Topologie neuronové sítě

Mezi nejznámější topologie umělých neuronových sítí patří vícevrstvá síť složená z vrstev neuronů (vstupní vrstva, jedna nebo více skrytých vrstev, výstupní vrstva). V rámci jedné vrstvy neexistují mezi neurony žádné vazby, ale neurony z jedné vrstvy jsou propojeny se všemi neurony sousední vrstvy (viz. obr. 2.2).

Předtím než začne trénování neuronové sítě, je třeba rozhodnout kolik bude mít struktura jednotek ve vstupní vrstvě, kolik jich bude ve skrytých vrstvách a kolik bude jednotek ve výstupní vrstvě. Normalizace vstupních hodnot každého atributu by měla pomoci zvýšit rychlost fáze učení. Typicky jsou vstupní hodnoty normalizovány tak, aby spadaly do intervalu 0 až 1. Diskrétní hodnoty atributů musí být zakódovány tak, aby byla jedna vstupní jednotka na jednu oblast hodnot. Neexistuje ale žádné předepsané pravidlo na počet jednotek ve skrytých vrstvách.

Návrh sítě je postupně se aproximující proces a může ovlivňovat přesnost výsledné trénované sítě. Inicializační hodnoty vah mohou ovlivnit výslednou přesnost.



Obrázek 2.2: Vícevrstvá topologie

2.9.4 Trénování neuronové sítě

U trénovacího procesu jde o co nejlepší nastavení vah. Cílem je použít takové trénovací soubory, aby se váhy na výstupu sítě blížily požadovanému výstupu. K nastavování jednotlivých vah se používá algoritmus back propagation [1], ale i jiné

dostupné techniky. Proces končí ve chvíli, kdy se dosáhne předem určené minimální chyby.

Back propagation

Back propagation [22], [4] je iterativní gradientní algoritmus učení neuronové sítě, který minimalizuje čtverce chybové funkce a který vykonává učení skrz vícevrstvou neuronovou síť. Je to opakované učení prostřednictvím nastavování vah. Během fáze učení se síť učí nastavovat váhy tak, aby byly schopny předpovídat správné třídy.

Algoritmus back propagation se skládá ze čtyř hlavních kroků: inicializace vah - váhy v síti jsou inicializovány malými náhodnými čísly (v rozsazích -1 až 1 nebo $-0,5$ až $0,5$), vytvoření trénovací množiny - dvojice vstup a žádaný výstup, výpočtu aktuálního výstupu a adaptace vah - učení. Prahy jsou také podobně jako váhy inicializovány malými náhodnými čísly.

Výpočet chyby

Počítání chyby zabraňuje rozdílu mezi aktuálním a nově vypočteným výsledkem. Chyba se zpátky posílá skrz neuronovou síť a váhy se tím přizpůsobují k minimalizaci další chyby.

2.9.5 Omezení neuronových sítí

Neuronové sítě jsou dobré pro předpovědi a odhady když:

- jsou vstupy dobře srozumitelné;
- je výstup dobře srozumitelný;
- jsou k dispozici zkušenosti – aplikace pro trénování neuronových sítí (expertní systémy).

Neuronové sítě jsou dobré pouze tak, jak jsou kvalitní trénovací algoritmy, které je učí (generují). Model neuronové sítě je statický, ale je třeba jej během času aktualizovat co nejvíce významnými příklady.

2.9.6 Výhody a nevýhody

Výhody neuronových sítí:

- schopnost vystihnout v datech nelineární vztahy;
- vysoká tolerance vůči šumovým datům;
- schopnost klasifikovat vzory, na které nebyly trénovány.

Nevýhody neuronových sítí:

- tendence příliš si přizpůsobovat data;
- není příliš zřejmé, co se uvnitř neuronové sítě děje – jednotlivé operace jsou hodně komplexní a těžko srozumitelné;
- často těžká interpretace výsledků.

2.10 Rozhodovací stromy

Rozhodovací stromy jsou jednoduché, rychlé a oblíbené metody pro třídění a předpovědi. Hlavním důvodem jejich oblíbenosti je, že hledají korelace ve velkých objemech dat novým, ale už ověřeným způsobem (ověřeno pomocí statistiky, inteligentních systémů apod.) a dají se jednoduše převést na klasifikační pravidla.

Model rozhodovacího stromu se skládá ze souboru norem pro dělení velkých různorodých uskupení na menší, více homogenní skupiny. Tím je myšleno, že se velké sbírky záznamů pomocí jednoduchých sekvencí rozhodování rozdělují do menších posloupných sad záznamů. Zmíněné sekvence roztřídí záznamy do odlišných skupin a větví prostřednictvím nejsilnější separace.

Vztahy, které rozhodovací stromy objeví v datech se používají pro odvození rozhodovacích pravidel, která jsou základem prediktivních modelů. Pomocí rozhodovacích stromů lze zjistit, které nezávislé proměnné mají největší vliv na cílovou proměnnou a poté je například aplikovat jako vstupní proměnné pro neuronovou síť. Při návrhu neuronové sítě se tedy použije jen minimální počet vstupních proměnných, což vede k minimalizaci typické vlastnosti neuronových sítí, kdy při velkém počtu vstupních proměnných nepodávají požadované výkony.

2.10.1 Dělení rozhodovacích stromů

Rozhodovací stromy se dělí podle počtu cest vedoucích z uzlu a podle funkce, ke které byly vytvořeny.

Dělení podle počtu cest vedoucích z uzlu:

- binární stromy – mají dvě cesty v každém dělení;
- trojité a vícecestné stromy – alespoň v jednom ze svých dělení mají tři a více cest.

Dělení podle funkce:

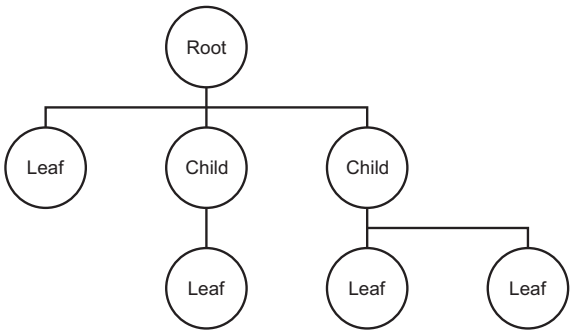
- klasifikační stromy – klasifikují objekty do tříd (v listových uzlech mají název třídy);
- regresní stromy – umožňují odhadnout hodnotu určitého numerického atributu (v listových uzlech mají konkrétní číselnou hodnotu).

2.10.2 Popis modelu rozhodovacího stromu

Uzly struktury rozhodovacího stromu dělíme na:

- kořenové uzly (root node) – vrchol nebo levý krajní uzel;
- potomky (child);
- koncové uzly (leaf node) – spodní nebo krajní pravý uzel.

Modely rozhodovacích stromů se kreslí zhora dolů (viz. obr. 2.3) nebo zleva doprava (viz obr. 2.4). Pravidlo rozhodovacího stromu je cesta z kořenového uzlu přes každý list.

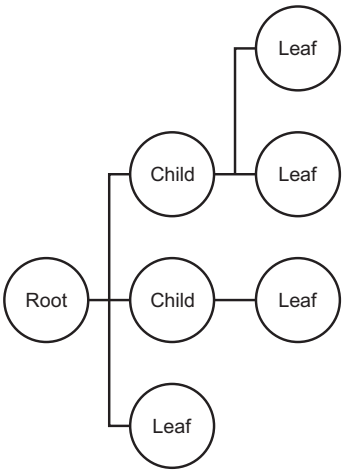


Obrázek 2.3: Model rozhodovacího stromu kreslený zhora dolů

2.10.3 Algoritmus rozhodovacího stromu

Při tvorbě rozhodovacího stromu se využívá rodina algoritmů TDIDT (Top Down Induction of Decision Trees), kdy se vstupní data rozdělují na menší a menší podmnožiny [9]. Do této rodiny algoritmů patří algoritmy ID3 [17], C4.5 [18], CART [23] a mnohé další. Postup algoritmu TDIDT:

1. volba jednoho atributu vstupní tabulky jako kořenu dílčího stromu;
2. rozdělení dat na patřičné podmnožiny a přidání uzlu pro každou z nich, vychází se přitom z hodnot vybraného atributu;



Obrázek 2.4: Model rozhodovacího stromu kreslený zleva doprava

3. pokud existuje uzel, jehož data nepatří do téže třídy, opakuje se u něj celý postup od bodu 1., pokud neexistuje, tak skončí.

Z toho lze usoudit, že růst stromu přestane, když všechny příklady uvažovaného uzlu spadají do téže třídy.

Při volbě atributu u jednotlivých uzlů se vychází z toho, že daný atribut musí od sebe rozdělit co nejvíce příkladů různých tříd. Jestliže budeme dělit data na menší části podle výsledků rozdělovacích kritérií, v ideálním případě bude každá část bez chyb (všechny data budou spadat pod danou část a budou náležet do stejné třídy). Mezi nejčastěji používaná rozdělovací kritéria patří: entropie, informační zisk, poměrný informační zisk a gini index.

Entropie

Entropie udává míru neuspořádanosti systému nebo míru neurčitosti procesu. Často se označuje jako Shannonova entropie a je definována:

$$H(A) = \sum_{i=1}^{n} (p(a_i) \log_2 p(a_i))$$

$$H(a_i) = -\sum_{i=1}^{n} (p(c_i \mid a_i) \log_2 (p(c_i \mid a_i)))$$

Kde A je atribut, a_i jsou hodnoty atributu, c_i jsou klasifikační třídy a n je počet tříd, do kterých je klasifikováno.

Informační zisk

Informační zisk je míra odvozená z entropie a používá se v algoritmu ID3. Logaritmická fukce o základu 2 je použita, protože je informace zakódována v bitech. Informační zisk je definován jako rozdíl mezi požadavky originální informace a novými požadavky (získanými po rozdělení). Jako rozdělovací atribut je brán atribut s největším informačním ziskem.

$$Zisk(A) = H(C) - H(A)$$

$$H(C) = -\sum_{i=1}^{n} (p(c_i) log_2(p(c_i)))$$

Poměrný informační zisk

Poměrný informační zisk je procento informačního zisku. Používá se proto, že entropie a informační zisk neberou v úvahu počet hodnot atributu. Informační zisk také preferuje výběr atributů, které mají velké číselné hodnoty, což není zrovna nejlepší řešení. Nevýhodou poměrného informačního zisku je, že se stává nestabilní při informačním rozdělení blížícím se 0. Atribut s nejvyšším poměrným informačním ziskem se vybírá jako rozdělovací atribut.

$$PIZ(A) = \frac{Zisk(A)}{Větvení(A)}$$

$$Větvení(A) = -\sum_{i=1}^{n} (p(a_i) \log_2 (p(a_i)))$$

Gini index

Gini index se používá u algoritmu CART a vyjadřuje číselný odklon Lorenzovy křivky od křivky dokonalého rozdělení. Bere v úvahu binární rozdělení všech atributů. Jestliže existuje v možných hodnot, pak existuje 2^v možných podmnožin. Atribut, který nejvíce redukuje špatná data, je vybrán jako rozdělovací atribut. Tento atribut představuje buď rozdělující podmnožinu (pro diskrétní hodnoty rozdělujícího atributu) nebo bod rozdělení (pro spojité hodnoty rozdělovacího atributu).

$$Gini = 1 - \sum_{i=1}^{n} (p(a_i)^2)$$

2.10.4 Prořezávání rozhodovacího stromu

Přořezávání rozhodovacích stromů má za úkol prořezávaný strom zmenšit a zjednodušit.

Pokud skončí větvení tím, že všechny příklady odpovídající jednotlivým listovým uzlům patří do téže třídy (provede se algoritmus TDIDT), může dojít k případu, že se požadavek na bezchybnou klasifikaci trénovacích dat přeučí. Jinak řečeno, ve struktuře se zohlední i závislosti, které jsou ve skutečnosti šumem a které souvisejí se samotnými trénovacími daty a nikoliv s analyzovanou problematikou.

Mimoto se také většinou stane, že je strom příliš rozvětvený (= malá srozumitelnost) a bezchybná klasifikace trénovacích dat není korektní (korekce u klasifikace je způsobena většinou šumem). Aby se předešlo těmto negativním vlivům, musí se zajistit, aby v listovém uzlu byly pokud možno příklady jedné třídy. Díky tomu se výsledný strom zmenší, zjednoduší a stane se srozumitelnějším, ale za cenu, že se u něj zhorší chování při klasifikaci trénovacích dat. Pokud je potřeba rozhodovací strom ořezat, lze to provést těmito způsoby:

1. modifikací původního algoritmu – nevytváří se původně myšlený, ale již ořezaný strom;

2. pozdějším prořezáním vzniklého úplného stromu.

V druhém případě se u jednotlivých nelistových uzlů posuzuje, jaké dopady bude mít odstranění daného uzlu na úplný strom.

Obecně pro oba případy platí, že schopnost prořezat strom znamená poznat, kdy se dá nelistový uzel nahradit listem. K této operaci se používají buď nová data (testuje se jimi jen požadované prořezání) nebo se provede statistický test z trénovacích dat, tzv. cross-validation [13]. Vyjádření algoritmu prořezání:

1. strom se převede na pravidla;

2. jednotlivá pravidla se odstraní z podmínky překladu a když dojde ke zlepšení odhadované správnosti, generalizuje se;

3. prořezaná pravidla se uspořádají podle odhadované správnosti a v této posloupnosti se použijí pro klasifikaci.

Více o této problematice lze nalézt v [9].

2.10.5 Výhody a nevýhody

Výhody rozhodovacích stromů:

- jsou snadno pochopitelné;

- mají schopnost snáze vysvětlit výsledek – snadné porozumění jejich výstupu (oproti ostatním algoritmům);
- jsou schopny zpracovávat jak číslicová tak i kategorická data;
- mají schopnost detekovat nelineární závislosti;
- dobrá intuitivnost jejich použití;
- dobře mapují sadu pravidel obchodování;
- dají se aplikovat na skutečné problémy.

Nevýhody rozhodovacích stromů:

- jsou omezené na jeden atribut výstupu (musí být navíc kategorický);
- algoritmy rozhodovacích stromů jsou nestabilní;
- stromy vytvořené z číselných datasetů mohou být složité.

3 NÁVRH ALGORITMU

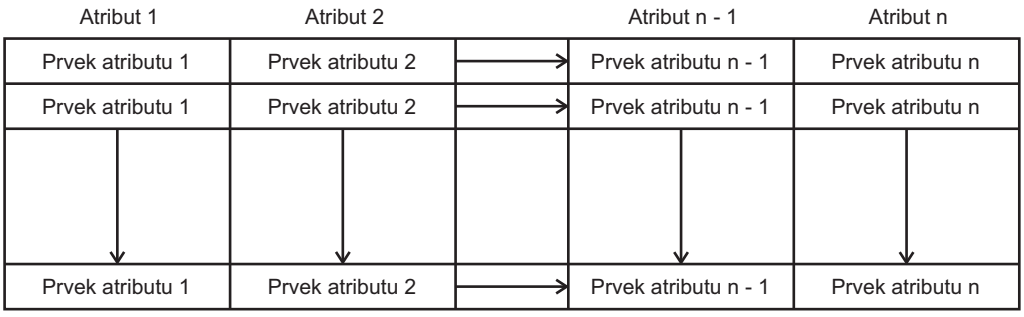
Algoritmus pro tvorbu rozhodovacího stromu byl implementován na platformě Java(TM) SE Development Kit. Jako vývojové prostředí byl použit program Eclipse.

3.1 Konstrukce algoritmu

Pro správné fungování rozhodovacího stromu je třeba mimo vytvoření správného algoritmu ošetřit vhodný výběr dat a jejich formát pro zpracování. Jde o to, že ještě předtím, než dojde k získání požadovaných informací pro vytvoření rozhodovacího stromu, je důležité pochopit a vhodně připravit (formátovat) vstupní data. Pokud se vychází z toho, že data před vstupem do dolovacího procesu vzešla z určitého dlouhodobého děje (sledované situace), většinou stačí mít správně popsány vstupní atributy (jednotlivé vlastnosti děje) a správně klasifikován a zařazen výstupní atribut.

Pokud se bude vycházet z toho, že data budou implementována prostřednictvím tabulky vstupních dat (viz. obr. 3.1), kdy budou atributy (sloupce tabulky) od atributu 1 po atribut $n-1$ představovat nezávislé proměnné a atribut n na nich závislou proměnnou (závislá, tzv. výstupní proměnná může být i v jiném sloupci, je to ale na úkor přehlednosti), je třeba brát ohled hlavně na to, aby atributy obsahovaly skutečnou vlastnost informace a stavy v jejich sloupcích jen některý z možných stavů. Při jakékoliv jiné formě dojde při extrakci požadovaných informací z těchto dat k chybě nebo zkreslení.

Příklady vstupních dat se nacházejí v tabulkách v sekci testování algoritmu (viz. tab. 3.1, tab. 3.2).



Obrázek 3.1: Tabulka vstupních dat

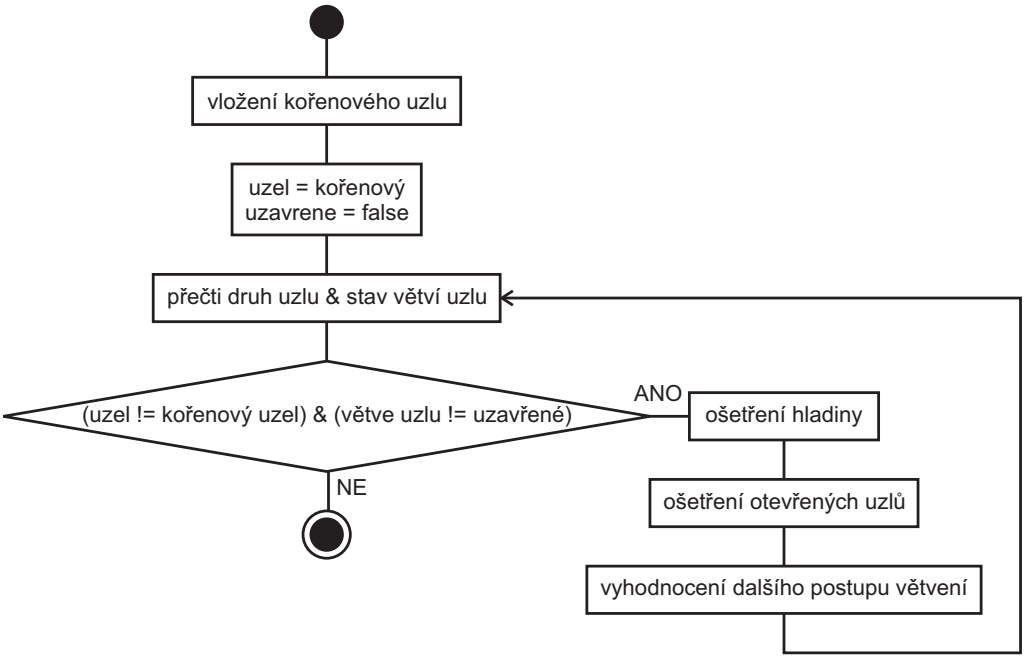
Princip fungování algoritmu je zobrazen na obrázku (viz. obr. 3.2). Nejprve se musí do rozhodovacího stromu vložit kořenový uzel, a poté, když je splněna

podmínka, že právě testovaný uzel není kořenový a zároveň nejsou všechny jeho větve uzavřené, se provedou tyto části:

1. ošetření hladiny;
2. ošetření otevřených uzlů;
3. vyhodnocení dalšího postupu větvení.

V opačném případě jsou všechna vstupní data ohodnocena, rozdělena do patřičných tříd (rozhodovací strom je vytvořený) a algoritmus se ukončí.

Popis jednotlivých dílčích částí algoritmu rozhodovacího stromu je uveden níže.



Obrázek 3.2: Vývojový diagram algoritmu pro tvorbu rozhodovacího stromu

3.1.1 Vlastnosti typů uzlů

V průběhu algoritmu se do stromu vkládají různé typy uzlů, které mají různé vlastnosti plynoucí z prováděného výpočtu. Uzly se rozdělují na tři hlavní kategorie: kořenový uzel, konečné uzly a otevřené uzly. Funkce, která tyto uzly do rozhodovacího stromu vkládá, je pro všechny kategorie uzlů stejná, rozdíl je pouze v množství informace, která je pro danou kategorii nezbytná.

Souhrn informací o uzlu:

- větve uzlu - ADT seznam řetězců *vetve uzlu* obsahující větve uzlu;
- cesta uzlů - ADT seznam řetězců *cesta uzlu* obsahující prošlé uzly od kořenového uzlu k počítanému uzlu;
- cesta stavů - ADT seznam řetězců *cesta stavu* obsahující prošlé stavy uzlů od kořenového uzlu k počítanému uzlu;
- prošlé větve - ADT seznam *prosle vetve* zahrnuje stavy *true* a *false* (stav *true* označuje, že je poduzel prošlý neboli uzavřený a stav *false*, že uzel ještě nebyl procházen a zřejmě také větven), během provádění algoritmu dochází u každého z uzlů k postupné změně stavu z *false* na *true*;
- vybraný index - celočíselný indikátor *vybrany index*, který slouží k zaznamenání, ve které větvi (popř. ve kterém poduzlu) se právě ohodnocení uzlu zastavilo.

Pro kořenový uzel se nastavuje název uzlu, větve uzlu, prošlé větve a vybraný index. Informace o cestě uzlů a cestě stavů musí zůstat prázdná, neboť kořenový uzel nemá žádného předchůdce a v případě jejich nastavení by hrozilo špatné rozdělení dat do tříd.

Konečné uzly obsahují pouze informaci o názvu uzlu. Tato informace ale narozdíl od ostatních dvou kategorií představuje závisle proměnnou ze vstupních dat algoritmu neboli jeden ze stavů, který obsahuje výsledný atribut (sloupec) ze vstupních dat.

U otevřeného uzlu se využívá všech výše uvedených informací. Jediným rozdílem je pouze různá doba jejich vkládání do uzlu zapřičiněná postupem celého algoritmu.

3.1.2 Typy navigací

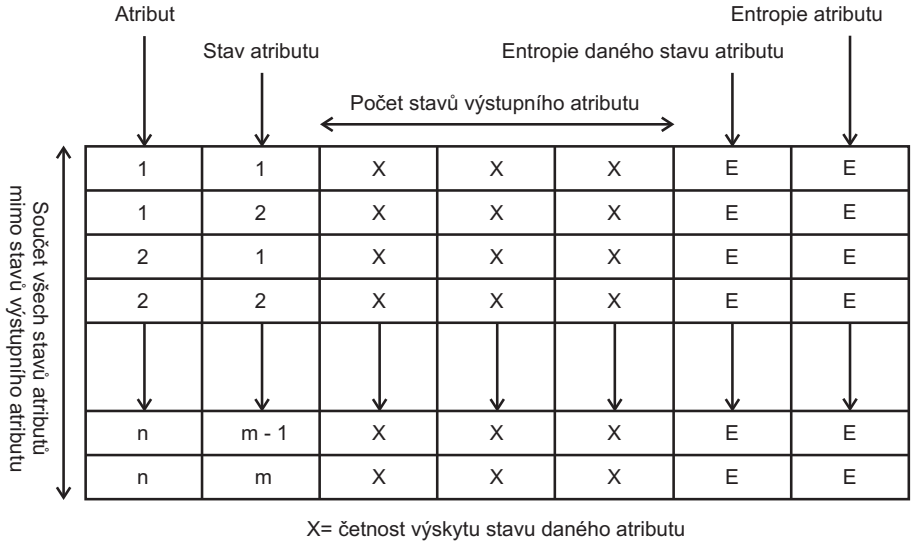
Navigace ve vytvořené stromové struktuře je realizovaná dvěma typy kroků:

- next - slouží pro skok z aktuálního uzlu na následující uzel; pokud je aktuální uzel otevřený, skáče se na jeho první větev zleva a pokud je konečný nebo uzavřený, skočí se na další větev téhož rodiče jako je aktuální uzel;
- top - při příkazu top se provede skok z aktuálního uzlu na jeho rodičovský uzel.

3.2 Dílčí části algoritmu

3.2.1 Vložení kořenového uzlu stromu

Parametry kořenového uzlu stromu se generují z extrahované tabulky *extr table* (viz. obr. 3.3) pro jejíž sestavení se využije všech vstupních dat. U kořenového uzlu je nutné nastavit indikátor prošlých větví *prosle vetve*, který poskytuje dvoustavovou informaci, jestli byla daná větev uzlu uzavřená nebo ne, a to na hodnotu *false*. Algoritmus je zobrazen na obrázku (viz. obr. 3.4).

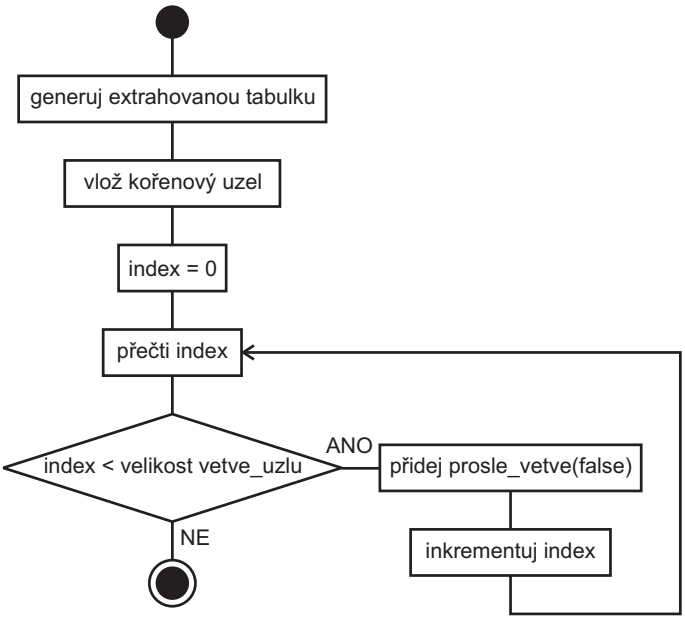


Obrázek 3.3: Extrahovaná tabulka - tabulka pro n atributů (atributy mimo výstupního atributu) a jejich m stavů

3.2.2 Ošetření hladiny

V části ošetření hladiny (viz. obr. 3.5) se nejdříve zjišťuje, jestli hodnoty větví uzlu *vetve uzlu* a konečné větve *konec vetve* generované extrahovanou tabulkou jsou stejné. Pokud stejné jsou, znamená to, že všechny větve uzlu budou ukončené a budeme vkládat pouze konečné uzly. Ještě je třeba ale zjistit, jestli se algoritmus nachází v kořenovém uzlu nebo ne. V případě, že v kořenovém uzlu bude, provede se vložení všech konečných uzlů a nastaví se hodnoty jejich indexů v *prosle vetve* (v nadřazeném uzlu) na hodnotu *true*. Kdyby algoritmus v kořenovém uzlu nebyl, provedou se stejné operace, ale poté je třeba ještě skočit na předcházející uzel a porovnat, jestli má také všechny *prosle vetve* v pořádku.

Když generované hodnoty *vetve uzlu* a *konec vetve* nejsou stejné, znamená to, že všechny větve uzlu nejsou konečné a je třeba vkládat uzly jeden po druhém. Proto se u jednotlivých větví určuje jestli jsou jejich uzly konečné nebo otevřené, vkládá



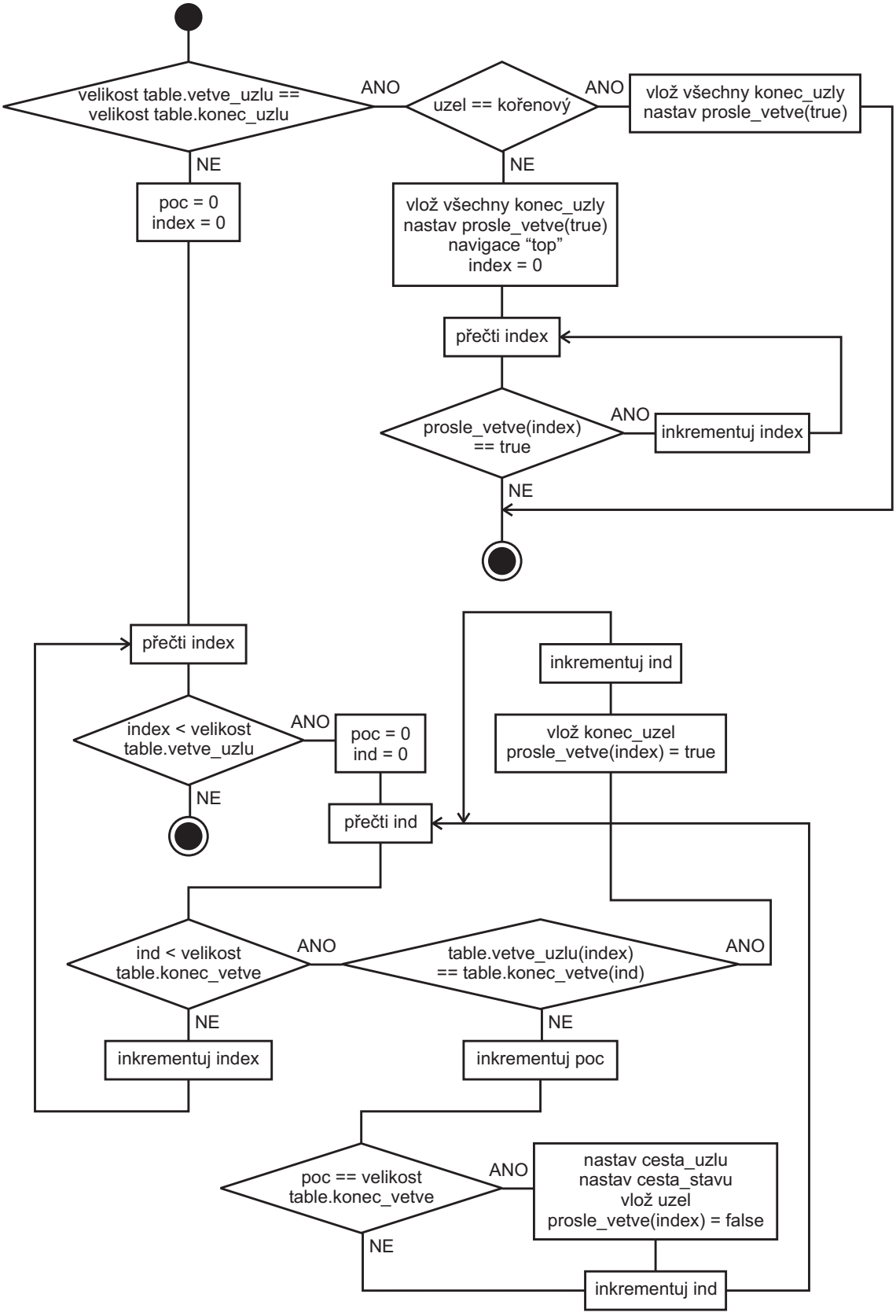
Obrázek 3.4: Vývojový diagram vložení kořenového uzlu stromu

se příslušný uzel a příslušná hodnota na příslušný index ADT stromu *prosle vetve* předchůdce. Rodíl mezi vložením ukončeného uzlu a otevřeného uzlu je v tom, že uzavřený uzel ponese jen svůj název, zatímco u otevřeného uzlu lze vložit pouze jeho *cestu uzlu* a *cestu stavu*, které musí být správně nastaveny. Je to z toho důvodu, že otevřený uzel se může dále větvit (generuje se u něj nová extrahovaná tabulka, určuje se název uzlu a všechny ostatní sounáležitosti jako u vkládání kořenového uzlu).

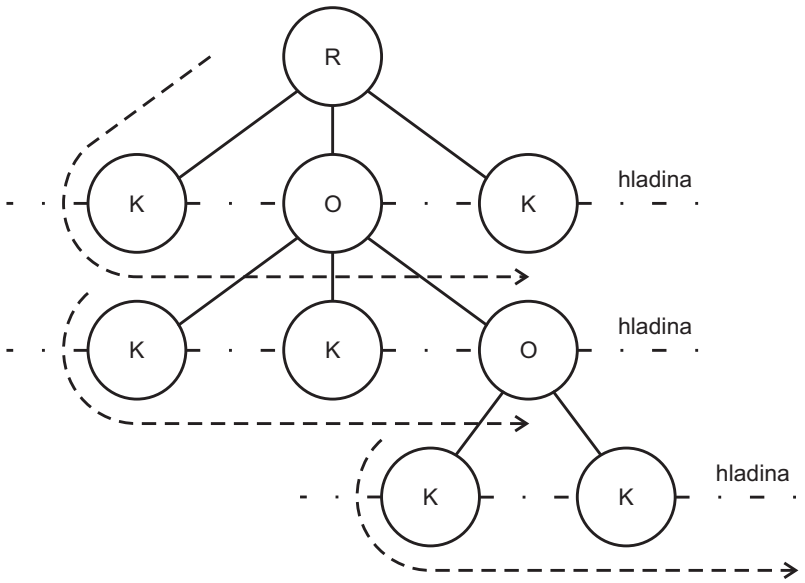
Grafická podoba hladin u rozhodovacího stromu je zobrazena na obrázku (viz. obr. 3.6).

3.2.3 Ošetření otevřených uzlů

Ošetření otevřených uzlů spočívá v tom, že se nejdříve v cyklu porovnávají stavy ADT seznamu *prosle vetve* s velikostí tohoto seznamu. Porovnávat se začíná vždy od hodnoty uzlu *vybrany index* (označuje, na kterém indexu ADT seznamu *prosle vetve* se algoritmus nachází, aby nedošlo k tomu, že se začne porovnávat pokaždé od stejné hodnoty, ale začne se od předchozí hodnoty a zjistí se patřičný přírůstek). Využívá se přitom také identifikátoru *falses*, který má na začátku cyklu nastavenou hodnotu *true* a při každém kladném porovnání, jestli je index ADT seznamu roven *true*, zůstane jeho hodnota na *true* a přírůstek *poc do* se inkrementuje. Pokud porovnaní není kladné (tj. dojde se na první otevřený uzel), změní se hodnota identifikátoru na *false* a žádné další navyšování přírůstku *poc do* už nenastane. Tímto způsobem se nalezne hodnota kolikrát je třeba zavolat navigaci na další uzel,



Obrázek 3.5: Vývojový diagram ošetření hladiny



Obrázek 3.6: Grafické zobrazení hladin u rozhodovacího stromu (R - kořenový uzel, K - konečný uzel, O - otevřený uzel)

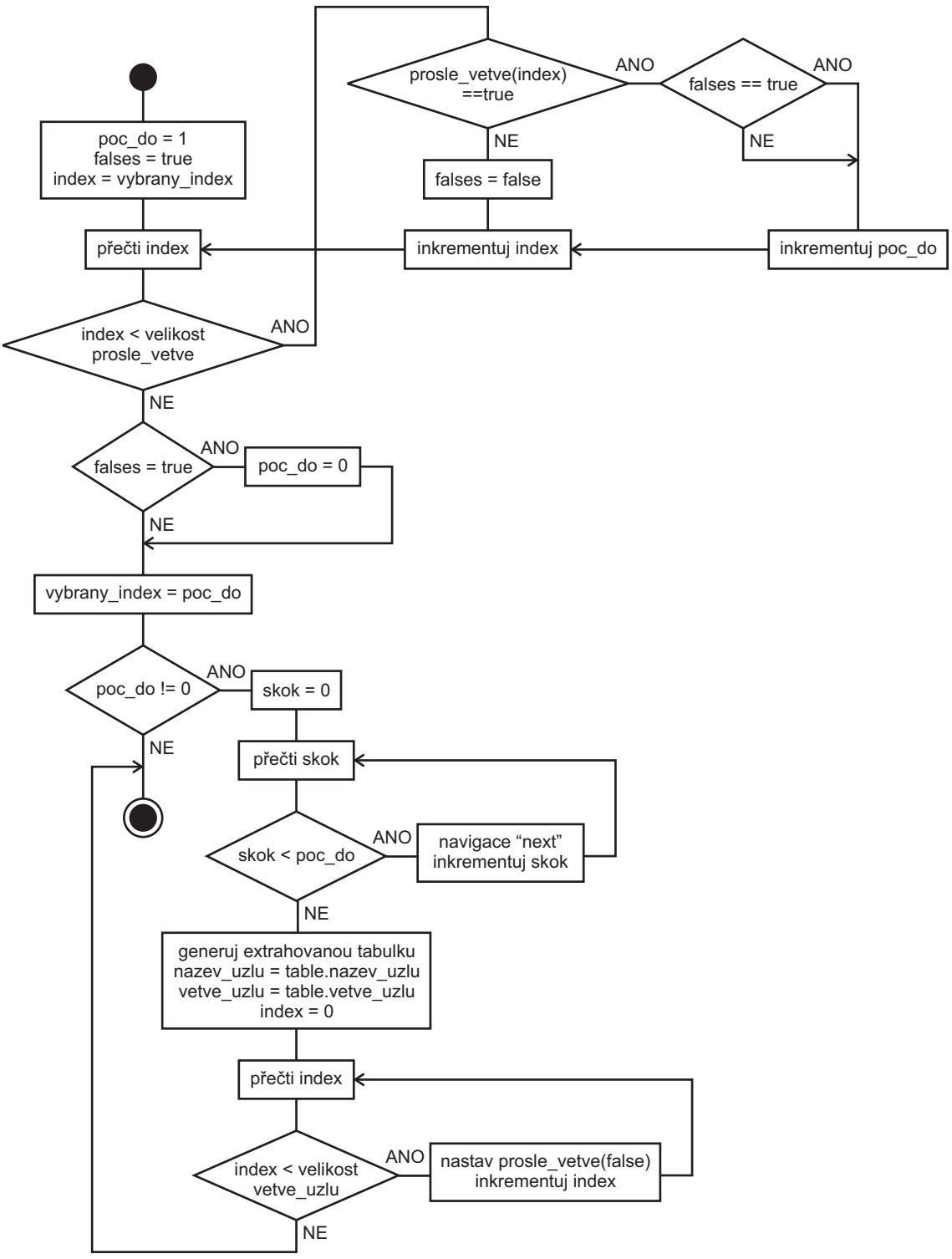
aby se docílilo prvního otevřeného uzlu a nebyl překonán rozsah navigace na další uzel.

Hodnota přírůstku se nastaví jako *vybrany index* uzlu a provede se přeskočení na první otevřený uzel, čímž se stane aktuálním uzlem pro další práci algoritmu. Ta spočívá ve vygenerování extrahované tabulky, která poskytne pro aktuální uzel jeho název a větve. Pro větve uzlu je pak potřeba v cyklu nastavit *prosle vetve* na hodnotu *false*. Vše je zobrazeno na obrázku (viz. obr. 3.7) Grafické zobrazení, jakým způsobem se prochází a vytváří celý rozhodovací strom je zobrazeno na obrázku (viz. obr. 3.8).

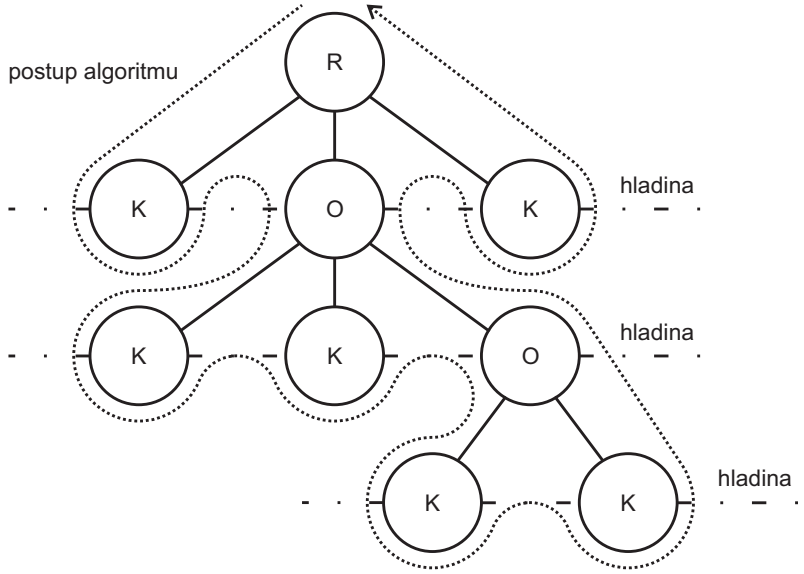
3.2.4 Vyhodnocení postupu větvení

Prvním úkolem této části je "přenést"aktuální uzel na co nejvyšší hladinu. Tím je myšleno, že pokud jsou všechny *prosle vetve* uzlu nastaveny na hodnotu *true* (uzel je uzavřen), skočí se pomocí navigace na uzel o hladinu výše (tj. na rodičovský uzel) a to se děje tak dlouho, dokud buď není algoritmus na kořenovém uzlu nebo pokud není tato podmínka vyvrácena.

Druhým úkolem je nastavit hodnotu identifikátoru *uzavrene* na hodnotu, která umožní provedení dalšího cyklu celého algoritmu (hodnota *false*), popřípadě algoritmus ukončí (při hodnotě *true*). Ukončení algoritmu je ale podmíněno navíc tím, že se aktuální uzel bude nacházet v kořenovém uzlu. Realizace nastavení identifikátoru *uzavrene* spočívá v tom, že se u uzlu projede ADT seznam *prosle vetve*



Obrázek 3.7: Vývojový diagram ošetření otevřených uzlů



Obrázek 3.8: Grafické zobrazení vytváření rozhodovacího stromu (R - kořenový uzel, K - koncový uzel, O - otevřený uzel)

a pokud má jednu hodnotu rovnu *false*, nastaví se identifikator *uzavrene* na hodnotu *false*.

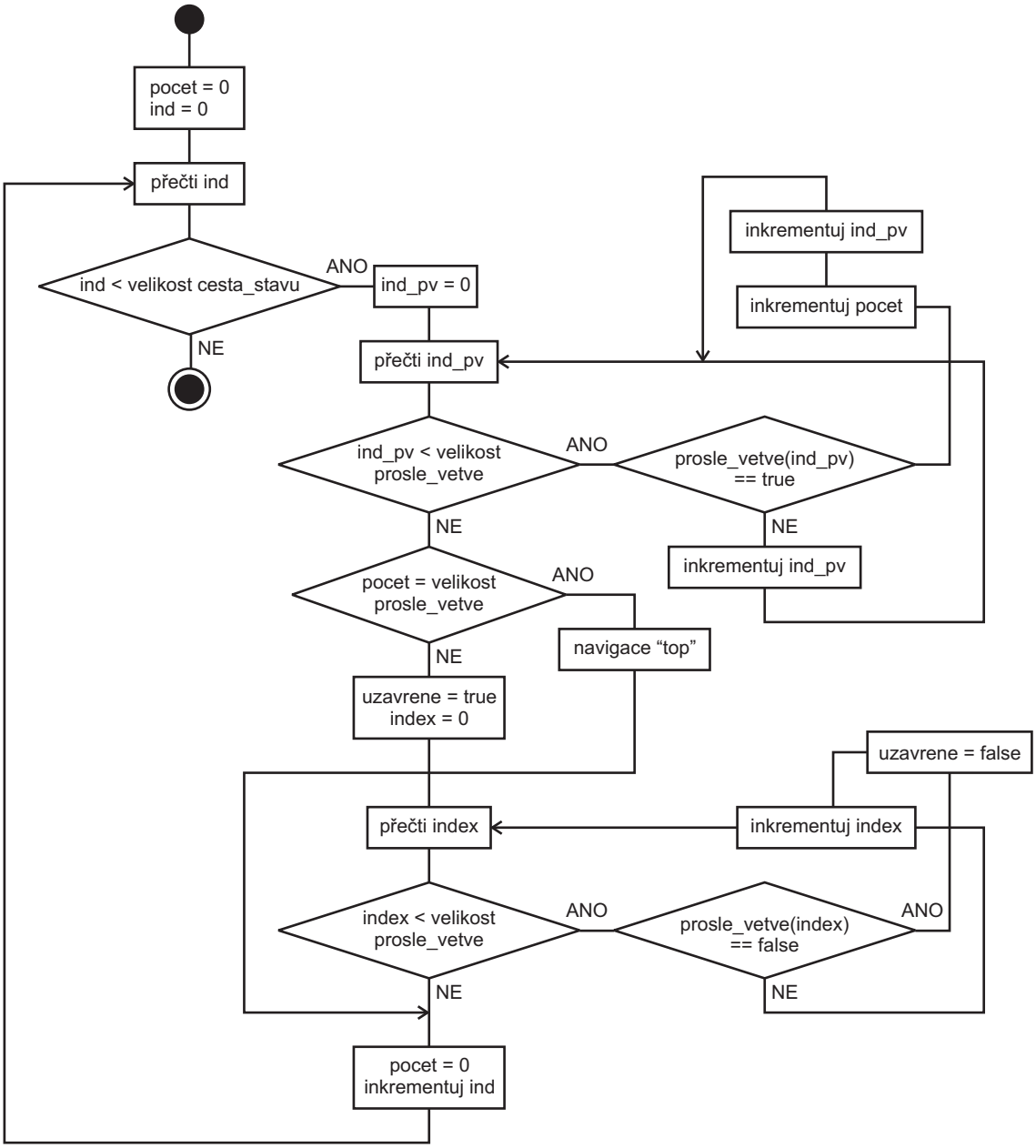
Oba úkoly jsou zachyceny na obrázku (viz. obr. 3.9).

3.2.5 Generování extrahované tabulky

Velikost extrahované tabulky (viz. obr. 3.3) definují její sloupce a řádky. Počet sloupců je určen dvěma vstupními sloupci (atribut, stav atributu), počtem prvků výstupního atributu a libovolným množstvím sloupců pro výpočet významnosti atributů (při použití entropie bylo využito dvou sloupců - entropie daného stavu atributu, entropie atributu). Počet řádků extrahované tabulky udává součet stavů atributů mimo stavů výstupního atributu.

Pro vygenerování extrahované tabulky je potřeba dodržet následující postup:

1. převod názvů atributů na čísla;
2. nalezení potřebných řádků;
3. zjištění hodnot prvních dvou sloupců a počtu hodnotících stavů;
4. vytvoření extrahované tabulky;
5. doplnění statistických hodnot;
6. výpočet entropie stavů atributů;



Obrázek 3.9: Vývojový diagram vyhodnocení postupu větvení

7. výpočet entropie atributů;
8. stanovení nejvýznamnějšího atributu a jeho vyhodnocení.

Podrobnější rozebrání bodů tohoto postupu je uvedeno níže.

Převod názvů atributů na čísla

Převod názvů atributů na čísla (viz. obr. 3.10) slouží k získání ADT seznamu *cisla atributu* a ADT seznamu *sloupce*. Vstupními parametry pro výpočet jsou jednorozměrné pole *atributy*, které obsahuje názvy atributů vstupních dat a ADT seznam *cesta uzlu*, ve kterém jsou uloženy názvy uzlů vedoucí k počítanému uzlu.

ADT seznam *cisla atributu* v sobě nese uložená jednotlivá čísla použitých vstupních atributů (jde o převedené názvy ze seznamu *cesta uzlu* do celočíselných hodnot). V algoritmu se porovnávají položky seznamu *cesta uzlu* s položkami seznamu *atributy* a v případě jejich shody se uloží číselná hodnota do seznamu *cisla atributu*.

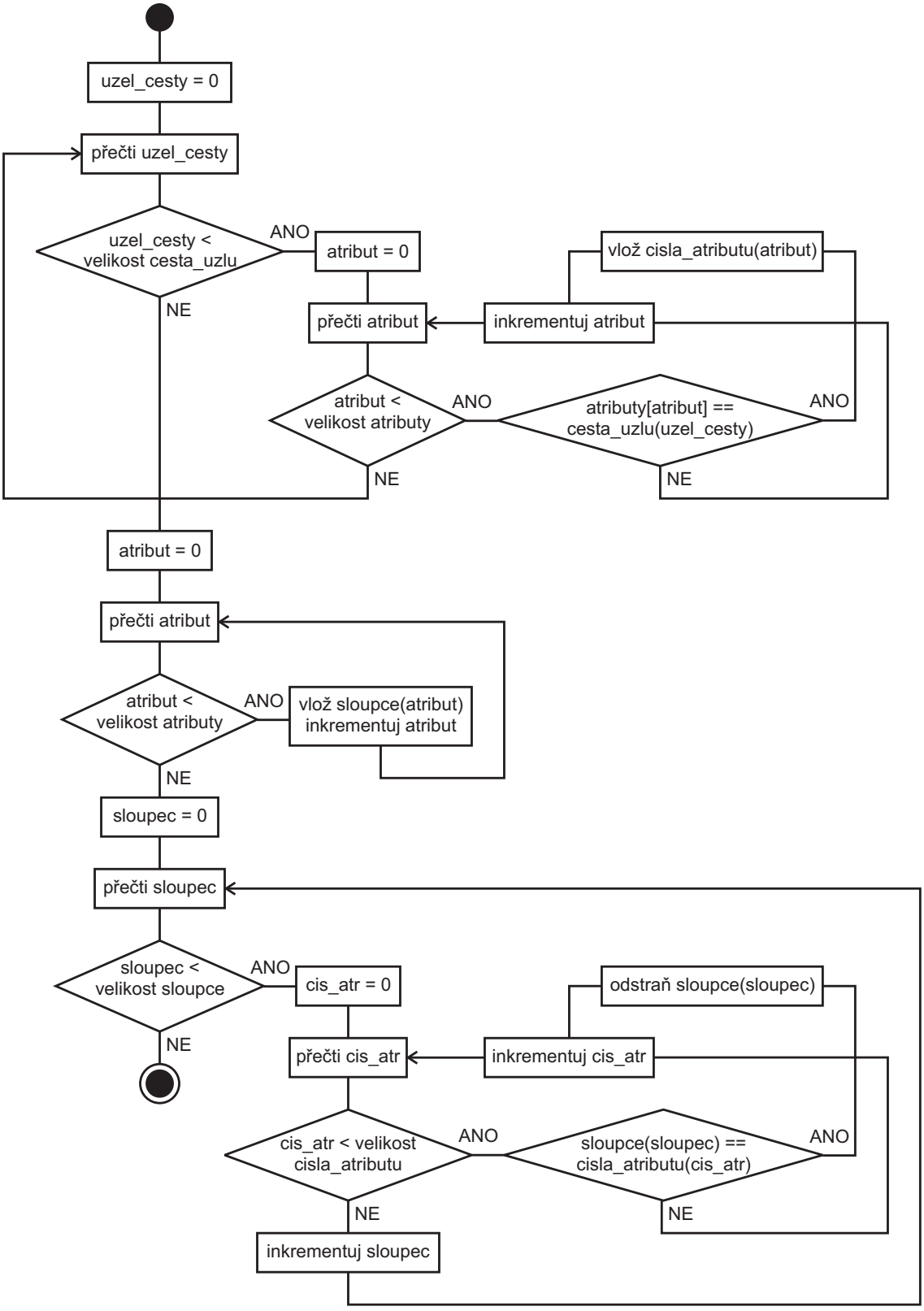
ADT seznam *sloupce* obsahuje celočíselné hodnoty atributů vstupních dat (sloupců vstupních dat) mimo těch atributů, které jsou obsaženy v seznamu *cesta uzlu* (těch už při vytváření extrahované tabulky není potřeba - vyplývá to z dělení dat do patřičných tříd). Nejprve se naplní ADT seznam *sloupce* hodnotami ADT seznamu *atributy* a poté, když se rovnají položky ADT seznamu *sloupce* položkám ADT seznamu *cisla atributu*, se provede v ADT seznamu *sloupce* jejich vymazání.

Nalezení potřebných řádků

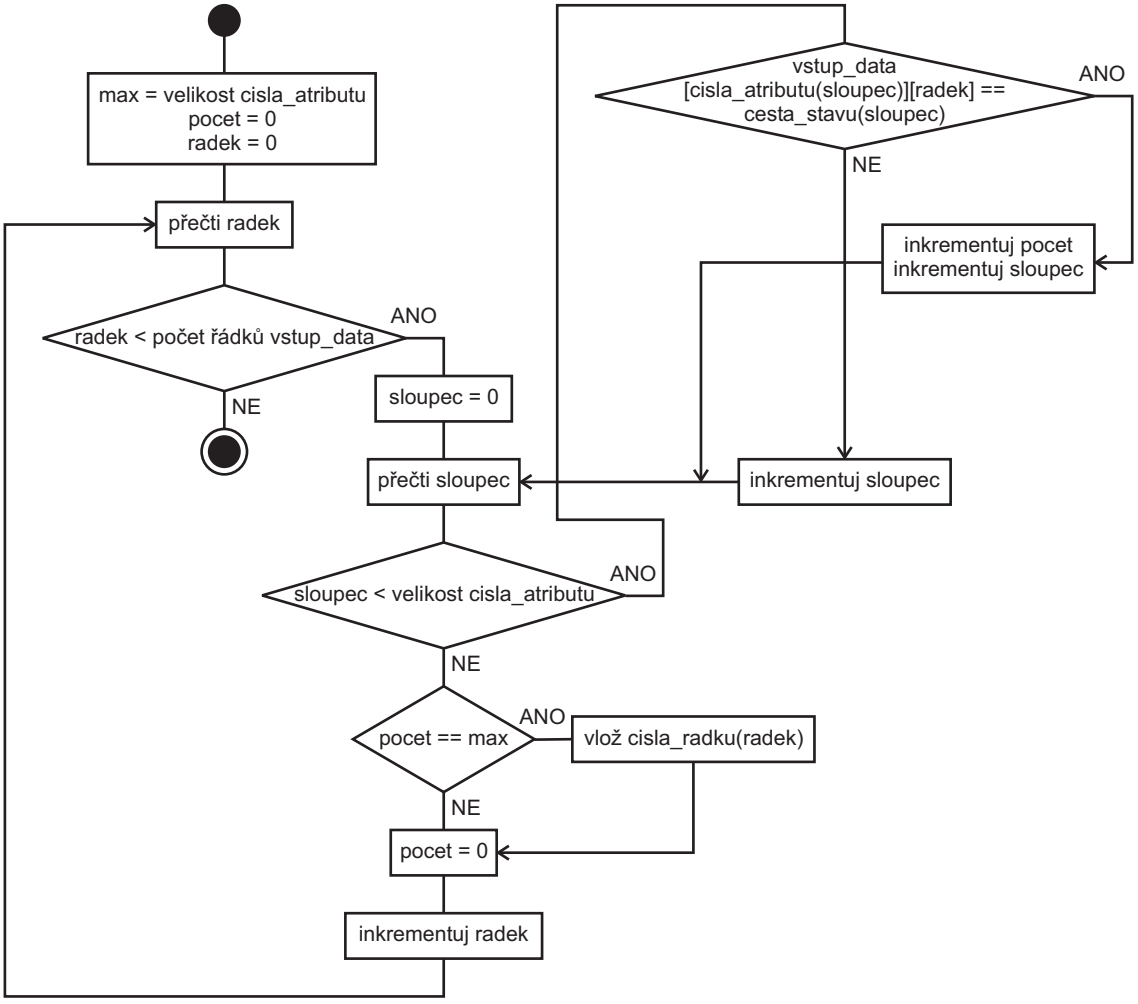
Funkce pro nalezení potřebných řádků má za úkol nalézt a zapsat řádky vstupních dat, které jsou nutné pro dělení do dané třídy podle entropie. Vývojový diagram této funkce je zobrazen na obrázku (viz. obr. 3.11) Vstupem jsou dvourozměrné pole vstupních dat *vstup data*, ADT seznam *cisla atributu* a ADT seznam *cesta stavu*, který obsahuje názvy stavů (větví) vedoucích k počítanému uzlu. Výpočet se provádí tak, že se u jednotlivých řádků vstupních dat prochází sloupce a porovnávají se jejich hodnoty s hodnotami v seznamu *cesta stavu*. Pokud se tyto hodnoty rovnají a rovnají se zároveň ve všech sloupcích současně, pokládá se řádek jako vyhovující a jeho číslo je vloženo do ADT seznamu *cisla radku*, který představuje výstup.

Zjištění hodnot prvních dvou sloupců a počtu hodnotících stavů

Zjištění hodnot prvních dvou sloupců u extrahované tabulky probíhá tak, že se prochází řádky u jednotlivých sloupců a porovnávájí se jejich hodnoty s hodnotami ADT seznamu *zasobnik* (slouží pro zjištění, jestli byl daný stav atributu



Obrázek 3.10: Vývojový diagram převodu názvů atributů na čísla



Obrázek 3.11: Vývojový diagram nalezení potřebných řádků

již vložen). Pokud jsou jejich hodnoty stejné, inkrementuje se celočíselná proměnná *vyskyt*. V dalším kroku se algoritmus ptá, jestli je proměnná *vyskyt* rovna nule. V případě že ano (to znamená, že daný stav atributu (pravě vybraného sloupce) ještě nebyl vložen do ADT seznamu *zasobnik*), je aktuální stav atributu přidán do ADT seznamu *zasobnik* a index atributu se přidá do ADT seznamu *atribut*, kde se ukládají atributy pro první sloupec extrahované tabulky. V případě, že je proměnná *vyskyt* různá od nuly, vynuluje se a nic se nepřidává. Dále se pak zaznamenávají počty stavů u jednotlivých atributů, a to do celočíselného ADT seznamu *pocty stavu* a všechny stavy daného atributu se přidají z ADT seznamu *zasobnik* do ADT seznamu *stavy atributu*. To se provádí pro každý sloupec vstupních dat. V ADT seznamu *stavy atributu* se nahromadí postupně všechny stavy. Na konci cyklu se pro každý sloupec zapíše do proměnné *poc hodn stavu* počet hodnotících stavů pro daný sloupec (tato proměnná se vždy pro každý atribut přepíše, a proto v ní zůstane jen počet hodnotících stavů pro poslední atribut, který obsahuje hodnotící stavy) a vynuluje se ADT seznam *zasobnik*. Graficky je celý průběh zobrazen na obrázku (viz. obr. 3.12).

Definování extrahované tabulky

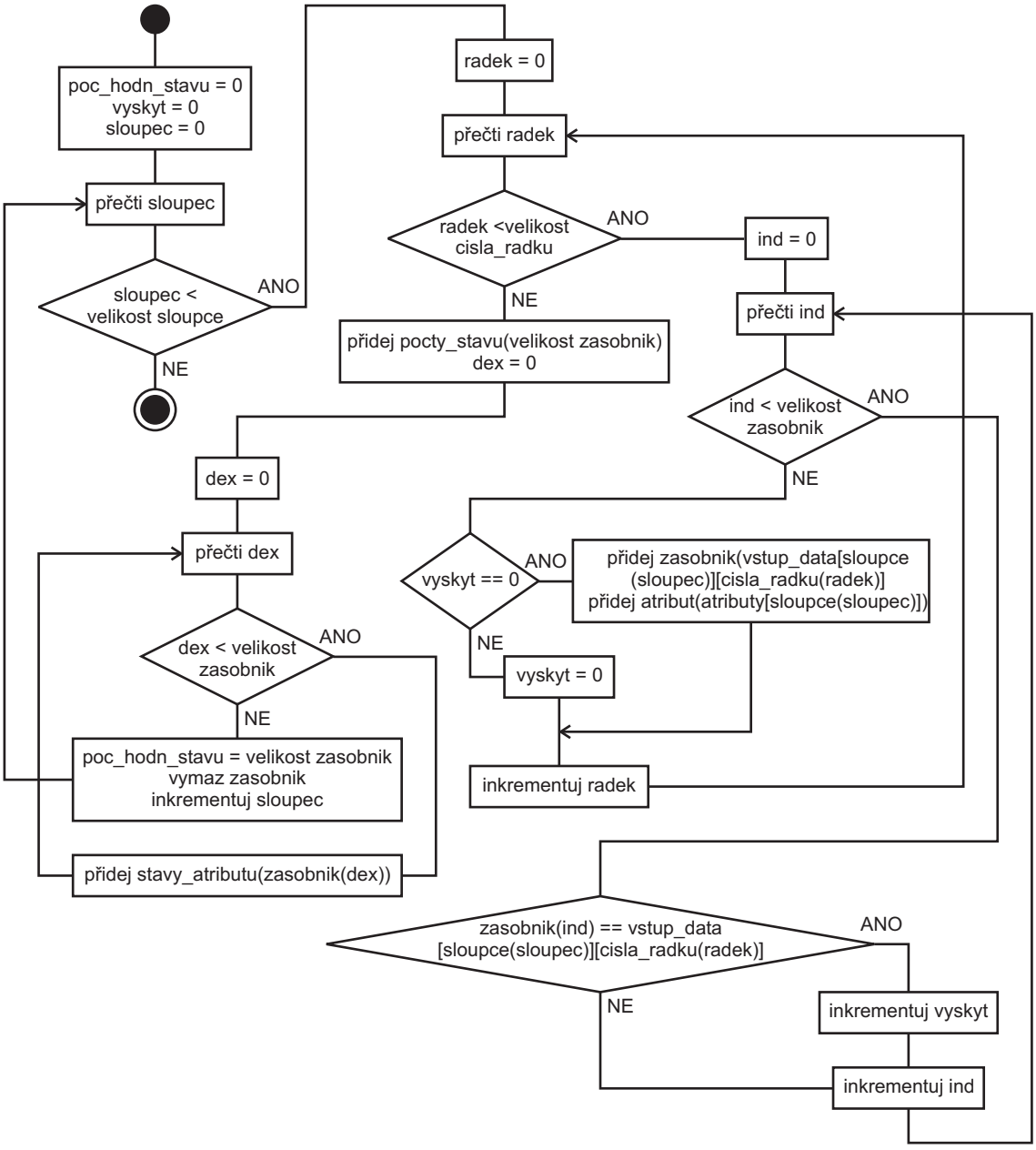
Blok vytvoření extrahované tabulky v sobě obsahuje kromě nadefinování velikosti extrahované tabulky také naplnění jejich prvních dvou sloupců, zaznamenání výstupních stavů a vymazání ADT seznamů, kterých již nebude potřeba (ADT seznam *stavy atributu* a ADT seznam *atribut*). Vše je zachyceno na obrázku (viz. obr. 3.13).

K naplnění prvních dvou sloupců stačí znát počet řádků extrahované tabulky. Pro každý řádek se vkládají do prvního sloupce extrahované tabulky hodnoty z ADT seznamu *atribut* a do druhého sloupce hodnoty z ADT seznamu *stavy atributu*. Tím jsou ošetřeny první dva sloupce. K zaznamenání výstupních stavů je použito ADT seznamu *vyst stavy*, do kterého se vloží patřičná data z ADT seznamu *stavy atributu* (vloží se jen ty stavy, které odpovídají poslednímu sloupci ve vstupních datech). Tyto hodnoty se posléze použijí v bloku doplnění statistických hodnot.

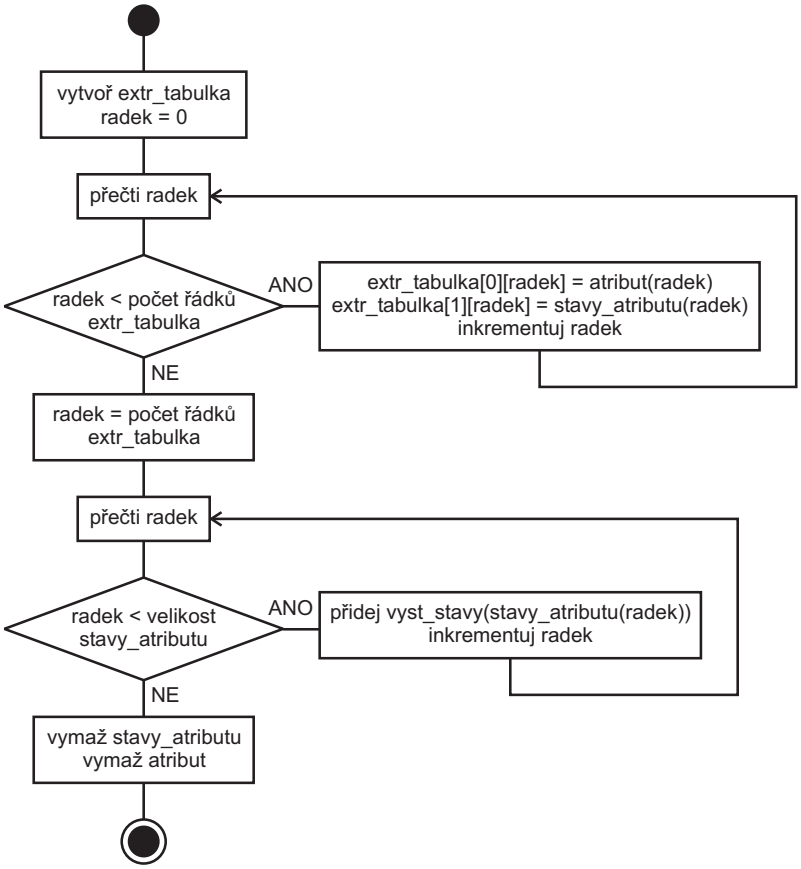
Doplnění statistických hodnot

Ještě než se začnou do extrahované tabulky doplňovat statistické hodnoty, vloží se do jejich polí nuly. Je to z toho důvodu, že se jednotlivé dané buňky extrahované tabulky postupně inkrementují. Navíc kdyby se neinkrementovaly, zůstane v nich hodnota nula místo prázdné hodnoty, kterou není třeba později ošetřovat.

Při samotném doplňování statistických hodnot jde o to, aby se hodnoty předem určených sloupců a řádků porovnávaly s patřičnými stavy atributů



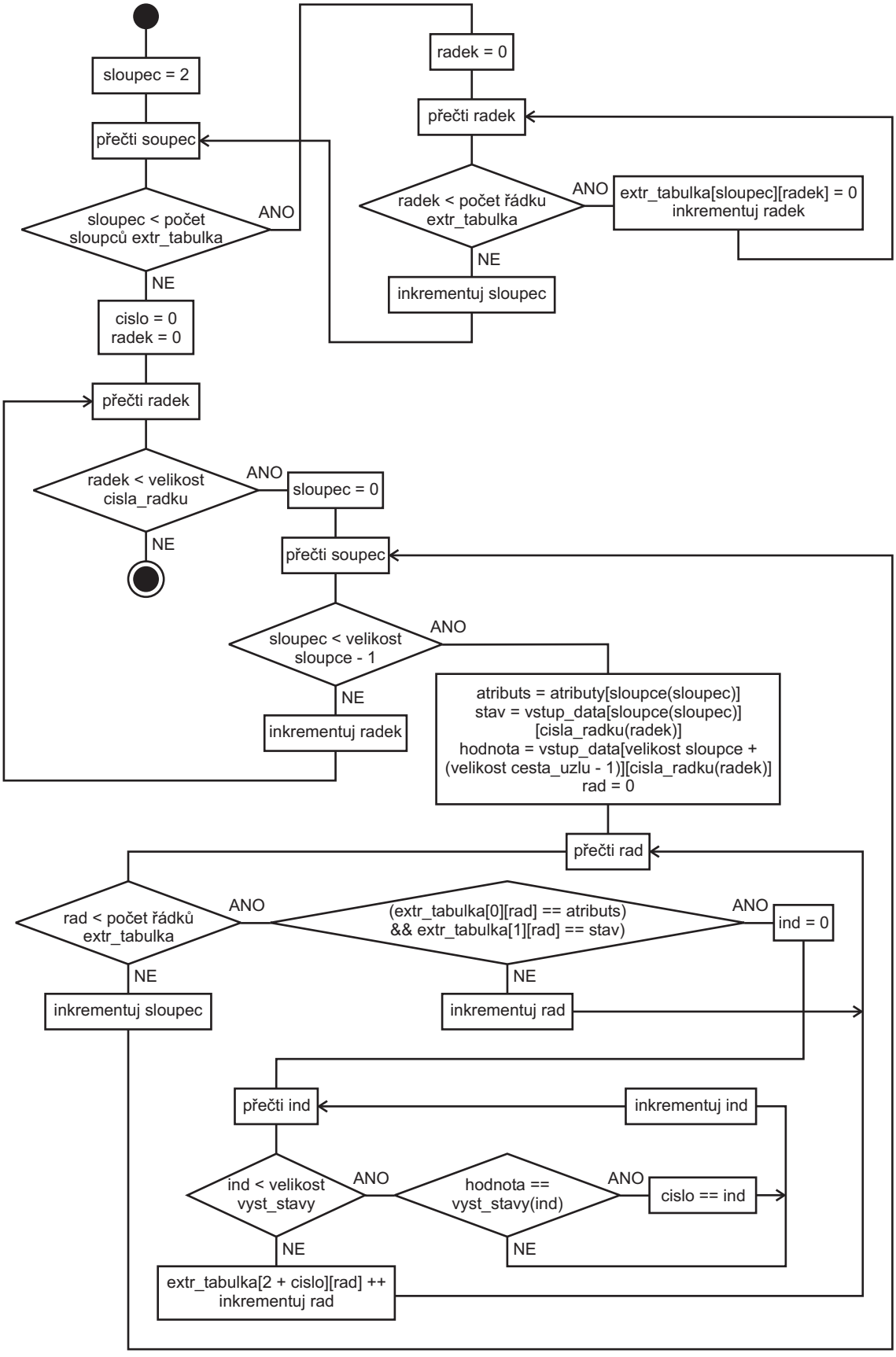
Obrázek 3.12: Vývojový diagram zjištění hodnot prvních dvou sloupců a počtu hodnotících stavů



Obrázek 3.13: Vývojový diagram definování extrahované tabulky

v závislosti na výstupních stavech a v případě jejich shody se v extrahované tabulce pod správnou buňkou inkrementovala její hodnota. Nejlépe je tento postup vidět na obrázku vývojového diagramu doplnění statistických hodnot (viz. obr. 3.14).

Funguje to tak, že se ve vstupních datech *vstup data* a jednorozměrném poli jejich atributů *atributy* prochází jednotlivé sloupce určeného řádku, zaznamenávají se název sloupce (do proměnné *atributs*), stav sloupce (do proměnné *stav*) a výstupní stav (do proměnné *hodnota*). Poté se pro určený řádek extrahované tabulky porovnává, jestli je shodný jeho název atributu s názvem atributu v proměnné *atributs*, a současně jestli je shodný název stavu atributu s názvem stavu atributu v proměnné *stav*, tím se nalezne řádek v extrahované tabulce s odpovídajícím atributem a stavem atributu. Pokud tomu tak je, projdou se jednotlivé indexy ADT seznamu *vyst stavy*, porovnají se z proměnnou *hodnota* a jsou-li stejné, uloží se číslo sloupce výstupního stavu do proměnné *cislo*. Tím jsou určeny všechny parametry buňky v extrahované tabulce a vybraná buňka se inkrementuje.



Obrázek 3.14: Vývojový diagram doplnění statistických hodnot

Výpočet entropie stavů atributů

Blok doplnění entropie stavů atributů představuje jakýsi mezivýpočet pro určení entropie atributů a počítá se jen ze statistických hodnot extrahované tabulky. Dvourozměrné pole vstupních dat *vstup data* a jednorozměrné pole jejich názvů atributů *atributy* nejsou zapotřebí.

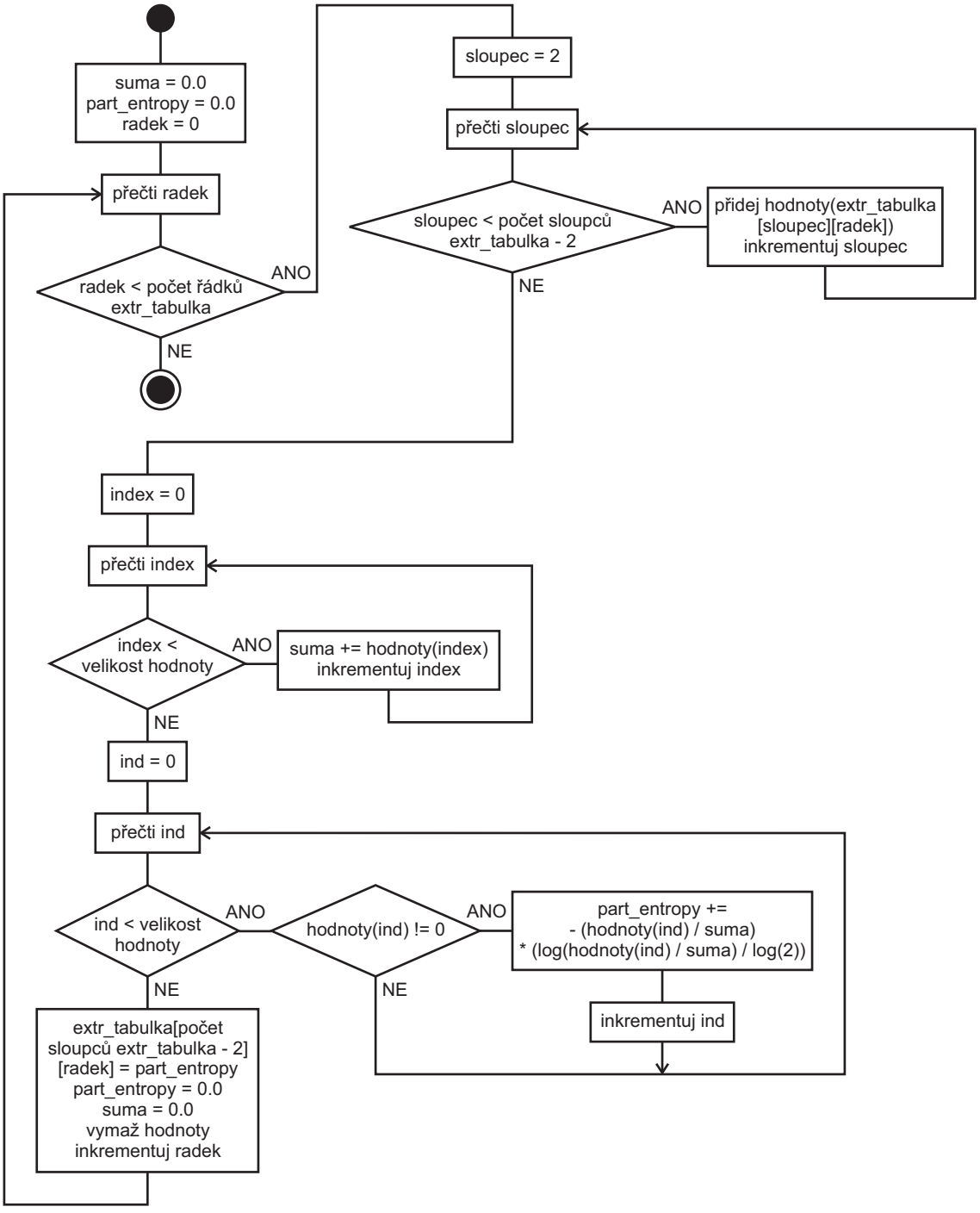
Pro určení hodnot entropie stavů atributů se pro každý řádek extrahované tabulky projdou sloupce obsahující statistické hodnoty extrahované tabulky (sloupce od indexu dva po počet sloupců extrahované tabulky bez dvou posledních. Ty představují právě entropii stavů atributů a entropii atributu). V prvním kroku se vloží hodnoty sloupců do ADT seznamu *hodnoty* a v druhém kroku se z nich vypočte suma. Třetí krok pro všechny hodnoty ADT seznamu *hodnoty* vypočítá do proměnné *part entropy* entropii. Vzorec výpočtu lze nalézt v teoretické části i v obrázku vývojového diagramu výpočtu entropie stavů atributů (viz. obr. 3.15). V poslední čtvrté části se hodnota proměnné *part entropy* vloží do příslušného řádku a sloupce extrahované tabulky, proměnné *part entropy* a *suma* se nastaví na nulu a vymaže se ADT seznam *hodnoty*.

Výpočet entropie atributů

Výpočet entropie atributů se provádí pro stejné řádky a sloupce extrahované tabulky jako výpočet entropie stavů atributů. Používá se i stejný ADT seznam *hodnoty*, který obsahuje stejné hodnoty daného řádku a opět se u něj provede výpočet sumy, která se uloží do proměnné *suma*. Rozdíl je v tom, že v dalším kroku se podle vzorce (viz. obr. 3.16 nebo teoretická část) vypočte entropie, která se dále přidá do ADT seznamu *entr*. Poté se vymaže ADT seznam *hodnoty* a proměnné *suma* a *entropie* se nastaví na nulu.

Protože hodnoty získané entropie představují jen dílčí hodnoty entropie atributů, které se pro jednotlivé atributy musí sečíst, je třeba v dalším kroku zajistit, aby se sčítaly vždy jen správné dílčí hodnoty entropie. To se provede pomocí celočíselné proměnné *atrib*, která představuje daný atribut a celočíselné proměnné *indikator*, která indikuje, kolik položek z ADT seznamu *entr* už bylo přičteno do proměnné *sum*. Celý cyklus pracuje tak, že se pro počet položek ADT seznamu *entr* přičte určená položka do proměnné *sum*, zvýší se proměnná *indikator* a pak se porovnává, jestli je proměnná *indikator* rovna položce ADT seznamu *pocty stavu* odpovídající zvolenému atributu. Pokud ano, přidá se *sum* do ADT seznamu *soucet* (ten shromažďuje entropie daných atributů), proměnná *sum* se nastaví na nulu, zvýší se proměnná *atrib* a proměnná *indikator* se nastaví na nulu.

Když jsou entropie atributu uloženy v ADT seznamu *soucet*, je jen zapotřebí je správně vložit do patřičných řádků. Pro tento krok se použije celočíselné proměnné



Obrázek 3.15: Vývojový diagram výpočtu entropie stavů atributů

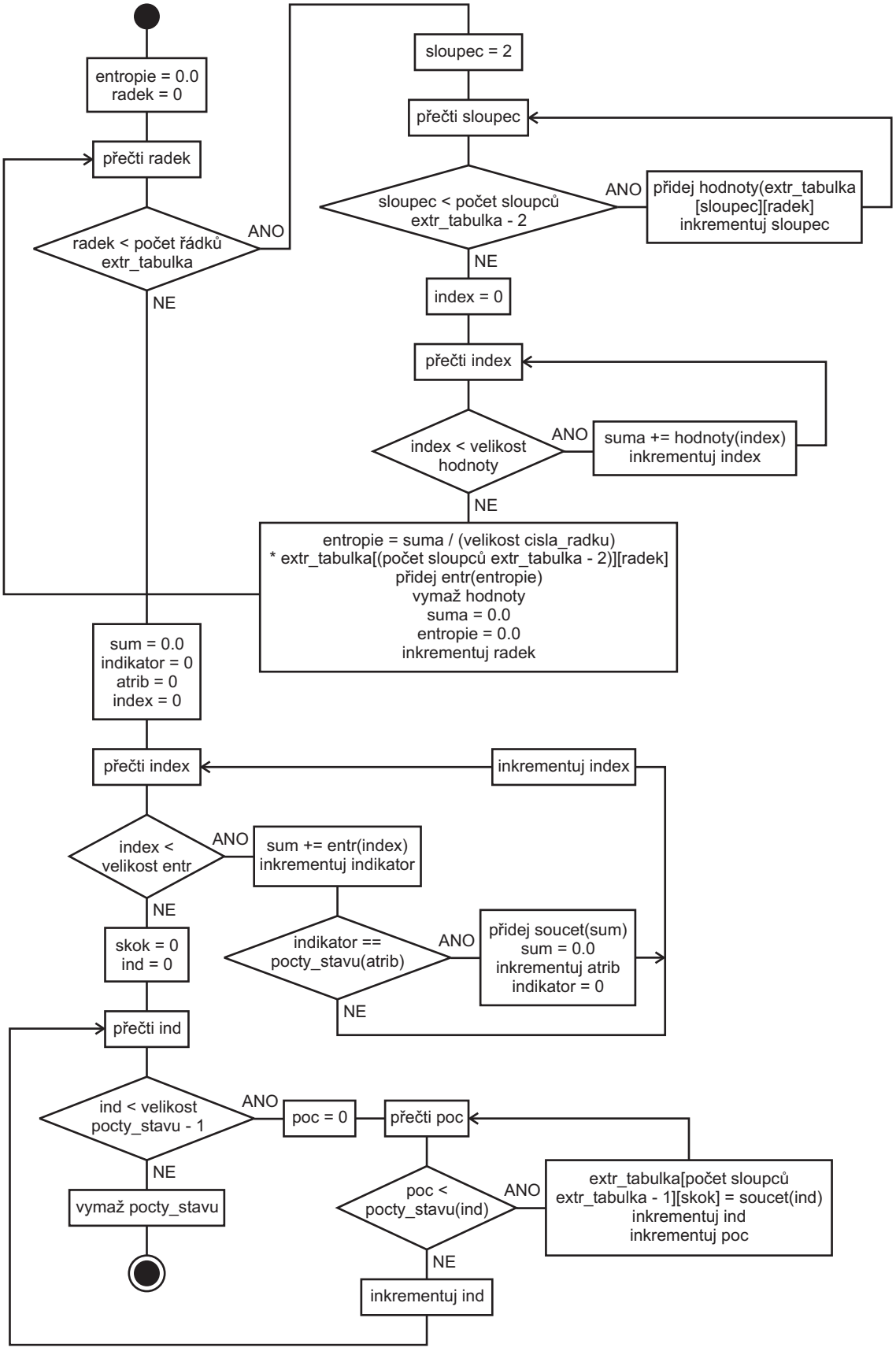
skok, která napoví, do kolika řádků se má daná hodnota z ADT seznamu *soucet* vložit. Pro každou položku mimo poslední (jde o položku výstupního stavu) v ADT seznamu *pocty stavu* se porovná, jestli už bylo dosaženo hodnoty v této položce a pokud ne, provede se vložení aktuální položky z ADT seznamu *soucet* (obsahuje hodnoty entropií atributů) do čísla řádku, kterou označuje proměnná *skok*, posledního sloupce extrahované tabulky. Pak se zvýší proměnná *skok* a přejde se na další atribut. Nakonec, když je celá extrahovaná tabulka hotová, se vymaže ADT seznam *pocty stavu*.

Stanovení nejvýznamnějšího atributu a jeho vyhodnocení

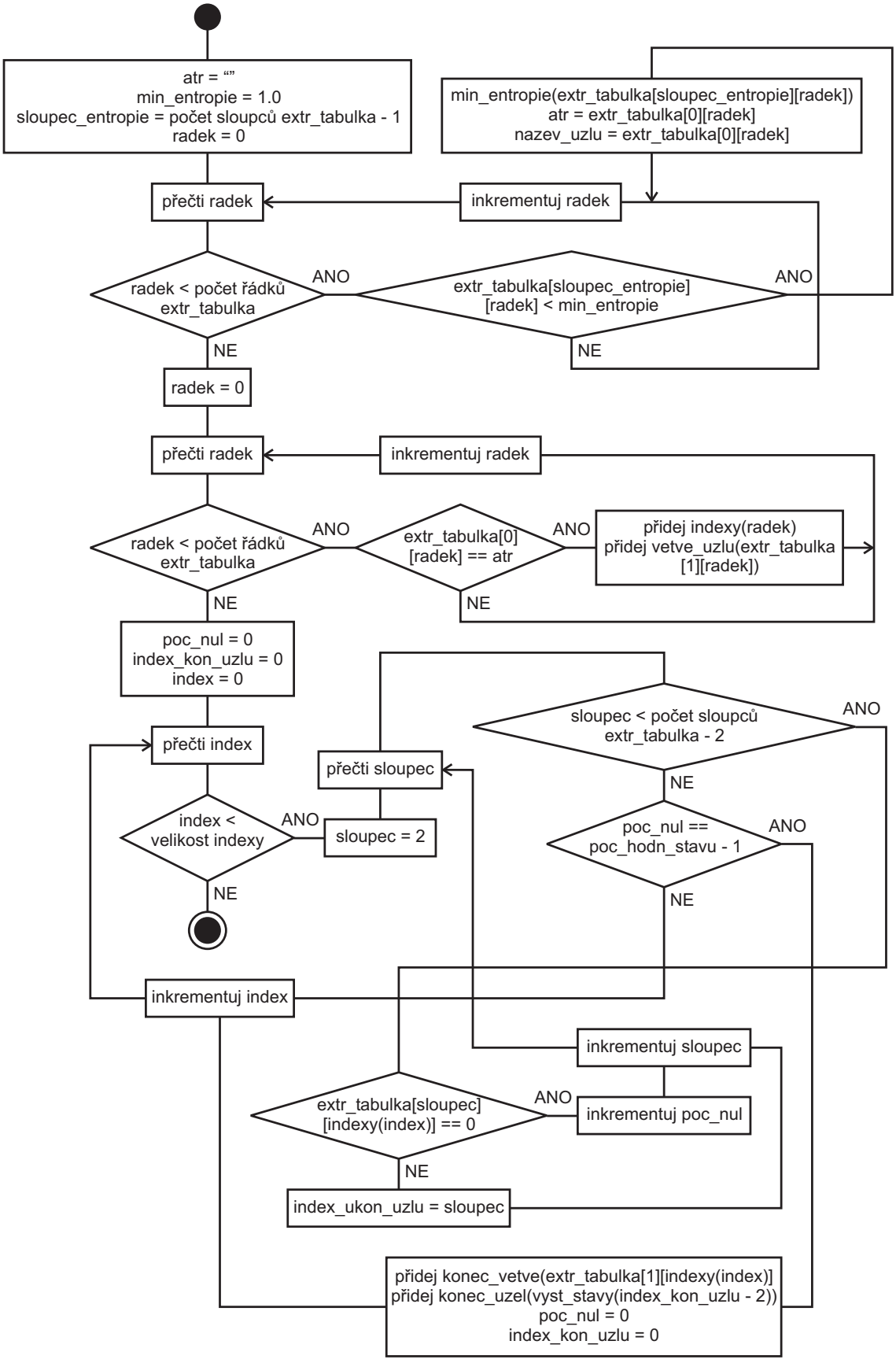
Tato část se skládá ze zjištění nejmenší hodnoty entropie (její atribut bude podle definice entropie brán za nejvýznamnější) a určení dat pro tvorbu uzlu. Vývojový diagram je zobrazen na obrázku (viz. obr. 3.17).

Při zjišťování nejmenší hodnoty entropie se využívá proměnné *min entropie*, jejíž hodnota se nastaví na jedna a poté se prochází poslední sloupec extrahované tabulky a hledá se entropie, která je menší než hodnota v proměnné *min entropie*. Když je menší hodnota nalezena, vloží se do proměnné *min entropie* a současně se do proměnné *nazev uzlu* uloží z prvního sloupce extrahované tabulky název uzlu. Ten se také uloží do proměnné *atr* pro další výpočet. Tímto postupem se vyhledala nejmenší hodnota entropie a atribut, který jí odpovídá.

Pro nalezení větví uzlu stačí projít u prvního sloupce všechny řádky extrahované tabulky, a když se rovnají proměnné *atr*, tak se hodnoty z druhého sloupce daného řádku přidají do ADT seznamu *vetve uzlu*. Navíc se pro určení případných konečných větví a konečných uzlů vloží číslo řádku do celočíselného ADT seznamu *indexy*. Když je tato část hotová, skočí se na část, která zjišťuje, jestli bude mít vkládaný uzel nějaké konečné větve a konečné uzly. V této části se pro každý řádek (každou položku ADT seznamu *indexy*) prochází sloupce extrahované tabulky od druhého po počet sloupců - 2 a ověřuje se, jestli není některá z hodnot rovna nule. Pokud ano, inkrementuje se celočíselná proměnná *poc nul* a pokud ne, nastaví se celočíselná proměnná *index kon uzlu* na hodnotu aktuálního sloupce. Pak se porovnává, jestli je hodnota proměnné *poc nul* rovna hodnotě proměnné *poc hodn stavu* - 1. Když tomu tak je, vloží se do ADT seznamu *konec vetve* hodnota extrahované tabulky z druhého sloupce a řádku odpovídající hodnotě z proměnné *index*. Mimoto se vloží i hodnota položky *index kon uzlu* - 2 z ADT seznamu *vyst stavy* do ADT seznamu *konec uzel*, proměnná *poc nul* a proměnná *index kon uzlu* se nastaví na nulu.



Obrázek 3.16: Vývojový diagram výpočtu entropie atributů



Obrázek 3.17: Vývojový diagram stanovení nejvýznamnějšího atributu a jeho vyhodnocení

3.3 Testování algoritmu

Funkčnost algoritmu pro vytvoření rozhodovacího stromu byla testována na testovacích datech stažených z internetových stránek, které navíc disponovaly grafickou podobou rozhodovacího stromu z daných dat. Díky tomu jde výsledek výstupu algoritmu lehce porovnat s výsledkem na daných internetových stránkách.

3.3.1 Test číslo 1

Z internetových stránek [16] pod položkou Chapter 6: Decision trees byla stažena prezentace se surovými daty (viz. tab. 3.1) pro vytvoření rozhodovacího stromu řešící oklasifikování chování počasí. Výsledná grafická podoba rozhodovacího stromu shodná s podobou v prezentaci je uvedena na obrázku (viz. obr. 3.18).

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
synny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

Tabulka 3.1: Tabulka surových dat (test číslo 1)

Vytvořená struktura

Algoritmus z poskytnutých surových dat vytvořil tuto strukturu:

```
Krok 1: root uzel: Outlook | vetve uzlu: [sunny, overcast, rain]
Krok 2: uzel pro vetveni: | cesta: [Outlook] [sunny]
Krok 3: konecny uzel: P
Krok 4: uzel pro vetveni: | cesta: [Outlook] [rain]
```

```
   Krok 5: * Humidity | vetve uzlu: [high, normal] cesta: [Outlook]
[sunny]
   Krok 6: konecny uzel: N
   Krok 7: konecny uzel: P
   Krok 8: * Windy | vetve uzlu: [false, true] cesta: [Outlook]
[rain]
   Krok 9: konecny uzel: P
   Krok 10: konecny uzel: N
```

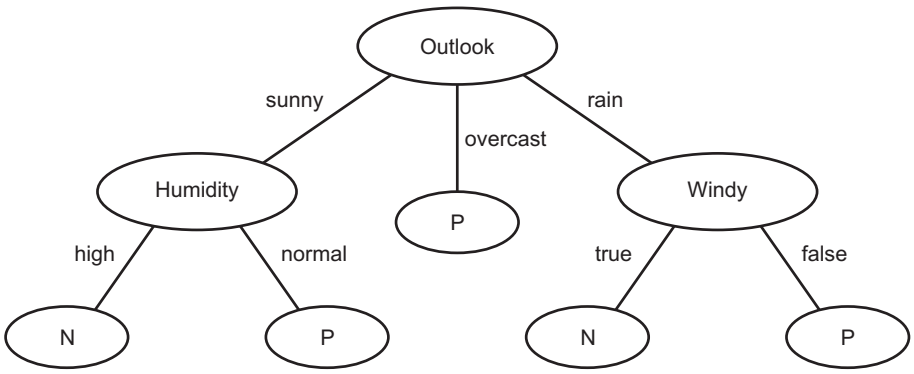
Popis vytvořené struktury

Krok 1 vytváří kořenový uzel Outlook, který má větve sunny, overcast a rain. V kroku 2 se vytvoří uzel, ke kterému vede cesta přes Outlook = sunny. Protože tento uzel je otevřený a bude se dále větvit, není ještě známý jeho název. Ve 3. kroku se vytváří konečný uzel (cestu k němu lze odvodit z kořenového uzlu - cesta je Outlook = overcast) a ve 4. kroku se opět vytváří otevřený uzel, který se bude dále větvit s cestou Outlook = rain. Kroky 2 až 4 tedy popisují ošetření hladiny (graficky viz. obr. 3.6) u kořenového uzlu.

Po zavedení kořenového uzlu a vytvoření uzlů jeho větví se skočí na uzel první větve kořenového uzlu. Extrahovaná tabulka určí, že jeho název bude Humidity, bude mít větve high a normal a cesta k němu bude Outlook = sunny. Tuto proceduru zobrazuje 5. krok. V 6. a 7. kroku se přiřadí větvím uzlu Humidity dva konečné uzly (pro větev high uzel N a pro větev normal uzel P), tím je uzel Humidity u první větve kořenového uzlu Outlook hotový a neboť je druhá větev kořenového uzlu ukončena konečným uzlem (ten byl vložen v kroku 3) přejde se na poslední třetí větev kořenového uzlu.

Krok 8 určil název uzlu třetí větve kořenového uzlu. Jde o atribut Windy, u kterého jsou větve uzlu false a true. Cesta pro identifikaci tohoto uzlu ve stromu je Outlook = rain. V 9. a 10. kroku se pro uzel Windy vloží dva konečné uzly (pro levou větev false to bude uzel P a pro pravou větev true to bude uzel N).

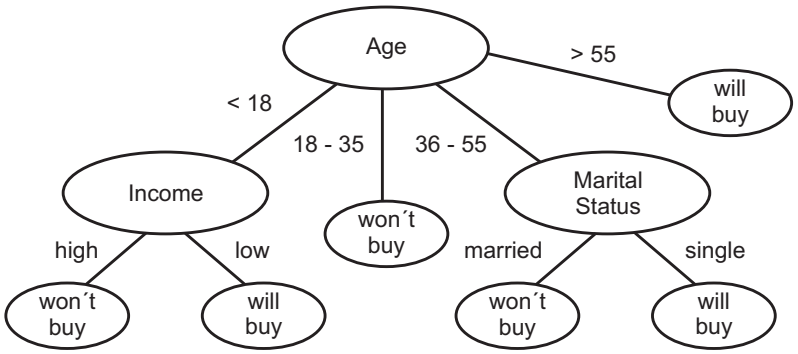
Těmito deseti kroky je rozhodovací strom z poskytnutých dat vytvořen a algoritmus se ukončí. Když se vytvořená struktura porovná s grafickou podobou rozhodovacího stromu nalezenou v prezentaci z internetových stránek (viz. obr. 3.18), je vidět, že vygenerovaný rozhodovací strom je stejný, rozdíl je pouze v přehození větví u uzlu Windy (ve vygenerovaném rozhodovacím stromu je vlevo větev false a vpravo větev true). Tento rozdíl je způsoben při vytváření extrahované tabulky, kdy se z poskytnutých dat (viz. tab. 3.1) u každého atributu ukládají stavy jak jdou po sobě (v daném sloupci zhora dolů). V tabulce tedy lze vidět, že u atributu Windy bude nejdříve stav false a až po něm stav true.



Obrázek 3.18: Rozhodovací strom (test číslo 1)

3.3.2 Test číslo 2

Test číslo 2 využívá dat z internetové stránky [19]. Data jsou zobrazena v tabulce (viz. tab. 3.2) a výsledná grafická podoba rozhodovacího stromu shodná s podobou na internetové stránce je zobrazena na obrázku (viz. obr. 3.19). Jde o rozhodovací strom řešící, zda si člověk v určitém věku, s určitým vzděláním a příjmem a v různém rodinném stavu koupí daný produkt.



Obrázek 3.19: Rozhodovací strom (test číslo 2)

Vytvořená struktura

Algoritmus z poskytnutých surových dat vytvořil tuto strukturu:

```
   Krok 1: root uzel: Age | vetve uzlu: [36-55, 18-35, < 18, > 55]
   Krok 2: uzel pro vetveni: | cesta: [Age] [36-55]
   Krok 3: konecny uzel: won't buy
   Krok 4: uzel pro vetveni: | cesta: [Age] [< 18]
   Krok 5: konecny uzel: will buy
   Krok 6: * Marital Status | vetve uzlu: [single, married] cesta:
[Age] [36-55]
```

Age	Education	Income	Marital Status	Purchase?
$36-55$	master's	high	single	will buy
$18-35$	high school	low	single	won't buy
$36-55$	master's	low	single	will buy
$18-35$	bachelor's	high	single	won't buy
<18	high school	low	single	will buy
$18-35$	bachelor's	high	married	won't buy
$36-55$	bachelor's	low	married	won't buy
>55	bachelor's	high	single	will buy
$36-55$	master's	low	married	won't buy
>55	master's	low	married	will buy
$36-55$	master's	high	single	will buy
>55	master's	high	single	will buy
<18	high school	high	single	won't buy
$36-55$	master's	low	single	will buy
$36-55$	high school	low	single	will buy
<18	high school	low	married	will buy
$18-35$	bachelor's	high	married	won't buy
>55	high school	high	married	will buy
>55	bachelor's	low	single	will buy
$36-55$	high school	high	married	won't buy

Tabulka 3.2: Tabulka surových dat (test číslo 2)

```
Krok 7: konecny uzel: will buy
Krok 8: konecny uzel: won't buy
Krok 9: * Income | vetve uzlu: [low, high] cesta: [Age] [< 18]
Krok 10: konecny uzel: will buy
Krok 11: konecny uzel: won't buy
```

Popis vytvořené struktury

Popis testu číslo 2 je obdobný jako u testu číslo 1. Ve výše uvedených 11 krocích se vygeneruje struktura rozhodovacího stromu, která je shodná s rozhodovacím stromem zdroje. Jak je patrné ze vstupních dat (viz. tab. 3.2), má vygenerovaný rozhodovací strom u uzlů Age, Marital Status a Income pouze přehozené pořadí větví, které je způsobené opět ukládáním stavů atributů při tvorbě extrahované tabulky. Nejde tedy o chybu, ale o rozdílný postup a pohled při vytváření algoritmu.

4 SYSTÉM PRO DOLOVÁNÍ INFORMACÍ Z BÁZE DAT NA ZÁKLADĚ GRID TECHNOLOGIÍ

Grid technologie je většinou hardwarová a softwarová infrastruktura poskytující standardizovaný a levný přístup k výpočetním službám. Je definována těmito body:

- koordinování zdrojů, které nepodléhají centralizované správě;
- používá standardní protokoly a rozhraní;
- poskytuje netriviální kvalitu služeb.

Jednoduše lze říci, že výhodou grid technologie je poskytnutí prostředků (výpočetní výkon, diskový prostor, přenosová kapacita sítě, speciální hardware apod.) subjektům, které je zrovna potřebují.

Jednoduchý systém pro dolování informací z báze dat, který byl zprovozněn na školním serveru, se skládá z upraveného algoritmu pro tvorbu rozhodovacího stromu a byl vytvořen na frameworku pro Javu GridGain [3].

4.1 Instalace

Instalace se rozděluje na část pro Javu a část pro GridGain.

4.1.1 Instalace Javy

Zprovoznění Java aplikace pro dolování informací z báze dat vyžaduje na serveru a pracovní stanici instalaci Javy. V tomto případě šlo o Sun Java(TM) Development Kit (JDK) 6 (balíček sun-java6-jdk). Instalace byla spuštěna příkazem `sudo apt-get install sun-java6-jdk`. Implicitně se tato verze Javy nainstaluje do /usr/lib/jvm/java-6-sun-1.6.0.07.

Možný problém při instalaci

Některé verze balíčků pro instalaci vyžadují během instalace jeden z dodatečných balíčků jdk-6-doc.zip nebo jdk-6-doc-ja.zip. Musí mít vlastníka root.root a musí se umístit do /tmp. Poté by s instalací Javy neměl být problém. Balíček jdk-6-doc.zip je přiložen u elektronické verze práce.

4.1.2 Instalace GridGain

Z domovské stránky projektu GridGain byla stažena nejnovější verze produktu (gridgain-unix-2.1.1.sh) a umístěna na server a pracovní stanici. Poté byla instalace na serveru spuštěna příkazem `sudo ./gridgain-unix-2.1.1.sh -c`. V případě stanice lze GridGain nainstalovat i v grafickém módu bez použití parametru `-c`. Implicitně se GridGain nainstaluje do /usr/local/gridgain 2.1.1.

4.2 Spuštění systému

Před spuštěním GridGain je nutné nastavit domovské cesty jak GridGain, tak i Javy. GridGain domovská cesta se nastaví na /usr/local/gridgain 2.1.1 a domovská cesta Javy na /usr/lib/jvm/java-6-sun-1.6.0.07. Použije se přitom příkazu `export JAVA HOME=/usr/lib/jvm/java-1.5.0-sun-1.5.0.16`. Pro GridGain pak `export GRIDGAIN HOME=/usr/local/gridgain 2.1.1`.

Ideální je tyto dva příkazy umístit do skupiny příkazů, které se spouštějí při přihlášení daného uživatele. Předejde se tím opětovnému definování obou domovských cest.

4.2.1 Spuštění node

Jednotlivé node se na serveru popřípadě stanici spouštějí v adresáři /usr/local/gridgain-2.1.1/bin příkazem `./gridgain.sh`.

4.2.2 Spuštění aplikace pro dolování informací z báze dat

Aplikace pro dolování informací z báze dat jsou na serveru umístěny v adresáři /home/condor/save/GridGain/data mining algorithms. Jsou zde umístěny dvě verze téže aplikace, a to v podsložkách 01 a 02. Obě mají shodný název dt.jar a byly vytvořeny z dříve uvedeného algoritmu pro tvorbu rozhodovacího stromu s použitím návodu k úpravě aplikace pracující na framework GridGain. Návod je možné nalézt na [3].

Spuštění aplikace se provede příkazem `java -jar dt.jar`. Po zadání tohoto příkazu se spustí aplikace dolování informací z báze dat. Data, která se prostřednictvím GridGain zpracovávají v aplikaci, pochází z algoritmu pro tvorbu rozhodovacího stromu - opět byla použita data pro vytvoření rozhodovacího stromu, zda si člověk v určitém věku, s určitým vzděláním a příjmem a v různém rodinném stavu koupí daný produkt.

4.2.3 Řešené problémy

GridGain běží implicitně pomocí multicastu. Jeho spuštění na školním serveru ale doprovázel problém neidentifikovatelných multicast adres. Multicast adresy by se přiřadily pomocí příkazu `sudo route add -net IPadresa netmask maskasite dev eth1`.

Další krok vedl k návštěvě správce sítě na Ústavu telekomunikací FEKT VUT v Brně a zjištění patřičných multicast adres. Po poradě se správcem sítě vyšlo najevo, že zprovoznění multicastu je vzhledem k vyšší politice jeho přidělení v rámci VUT v Brně časově náročnější a musí projít schvalovacím procesem. Proto bylo nutné najít náhradní řešení.

Náhradní řešení představovalo zprovoznění GridGain pomocí JGroups TCP, kdy se multicast nahradí TCP spojením mezi danými uzly.

Dalším důležitým bodem pro spuštění GridGain bylo vypnutí IPv6 a nastavení použitých IP adres do souboru /etc/hosts.

Po všech těchto krocích již šel GridGain na serveru spustit a byla na něm zprovozněna aplikace pro dolování informací z báze dat (viz. výše). Vytvořený rozhodovací strom odpovídal vytvořenému rozhodovacímu stromu ze sekce testování algoritmu. Jediné co nešlo, bylo vzájemné provázání serveru a pracovní stanice a jejich současný podíl na výpočtu, výpočet prováděl pouze server.

Konfigurace JGroups TCP a pomocná nastavení

JGroups TCP jsou součástí GridGain a jeho konfigurační soubory se nacházejí v /usr/local/gridgain-2.1.1/config/jgroups v podadresářích multicast (pro konfiguraci multicastu) a tcp (pro konfiguraci TCP). Při zprovozňování JGroups TCP byl důležitý podadresář tcp s konfiguračním souborem spring-jgroups.xml.

V souboru spring-jgroups.xml je důležité nastavit cestu k hlavnímu konfiguračnímu souboru JGroups TCP s názvem jgroups.xml. Umístění jgroups.xml bylo zvoleno do /home/condor/config a byl nastaven jak podle návodu na [3], tak i z dalších různorodých zdrojů nalezených na internetu.

Vypnutí IPv6

Verze Ubuntu serveru na školním serveru nemá narozdíl od novějších verzí Ubuntu serveru integrovaný protokol IPv6 přímo v jádře. Proto je jeho vypnutí o dost jednodušší. Spočívá v konfiguraci souboru /etc/modprobe.d/aliases, kde je nutné nastavit položku `alias net-pf-10 off`. Další položkou, která musí být změněna je `alias ipv6 off`. Tím se zakáže IPv6 v jádře a je třeba provést update modulů příkazem `update-modules`. V případě, že tento příkaz vyhodí chybovou hlášku,

lze použít příkazu `sudo dpkg --configure -a`. Poté již jen stačí systém restartovat (rebootovat).

ZÁVĚR

Cílem této práce bylo seznámit se s technikami pro klasifikaci a predikci dat a jejich možnostmi. Práce se snaží poskytnout ucelený pohled na tyto techniky se zaměřením na nejčastěji používané metody.

První část práce přináší obecný popis klasifikace a predikce, přibližuje klasifikační proces a přípravu dat pro jeho vstup. Mezi vybrané metody klasifikace byly zahrnuty asociační pravidla, Bayesovské klasifikace, genetické algoritmy, metoda nejbližšího souseda, neuronové sítě a rozhodovací stromy, které byly do nejmenšího detailu popsány. Dále byly v práci popsány metody predikce, kde patří lineární a nelineární regrese.

V druhé části byl podrobně vysvětlen algoritmus pro tvorbu rozhodovacího stromu. Jednotlivé dílčí části algoritmu jsou co nejvíce rozebrány a jsou k nim připojeny dané vývojové diagramy. Ke konci druhé části je u algoritmu pro tvorbu rozhodovacího stromu provedeno testování. Funkčnost byla testována na vzorku dat, které měly navíc v příloze uvedenou grafickou podobu rozhodovacího stromu, tím bylo zaručeno snadné porovnání. V obou dvou testech byly vygenerované struktury rozhodovacích stromů totožné s grafickými podobami rozhodovacích stromů z internetových stránek a lze říci, že algoritmus pro tvorbu rozhodovacího stromu pracuje správně.

Třetí část práce popisuje vytvoření a zprovoznění výpočetního gridu, který lze dále rozšiřovat a lze s jeho pomocí realizovat i vysoce náročné výpočty v krátkém čase. Je zde uvedena jak instalace grid technologie GridGain, tak i její spuštění a řešení vzniklých problémů. U řešení vzniklých problémů je uveden krátky obecný postup k nalezení řešení. Samotná aplikace, která byla na frameworku GridGain zprovozněna, vychází z upraveného algoritmu popsaného v druhé části práce. Vygenerovaná struktura po spuštění GridGain odpovídá struktuře z poskytnutých dat.

REFERENCE

[1] Backpropagation: Theory, Architectures, and Applications (Paperback). In *Backpropagation: Theory, Architectures, and Applications (Paperback)*, 1995, ISBN 978-0805812596, str. 576.

[2] Data mining: jak z vašich dat vytěžit maximum: sborník k seminářům. In *Data mining: jak z vašich dat vytěžit maximum: sborník k seminářům.*, Praha, 2002, ISBN 80-238-9408-0, str. 112.

[3] GridGain Systems: GridGain Open Cloud Platform. 2005-2009.
URL `http://gridgain.com/`

[4] The Backpropagation Algorithm. In *The Backpropagation Algorithm*, 2007.
URL `http://page.mi.fu-berlin.de/rojas/neural/chapter/K7.pdf`

[5] Berka, P.: Dobývání znalostí z databází. In *Dobývání znalostí z databází*, Praha, 2003, ISBN 80-200-1062-9, str. 366.

[6] Berry, M.; Linoff, G.: Companion Pages for Data Mining Techniques for Marketing, Sales, and Customer Relationship Management (Second Edition). In *Companion Pages for Data Mining Techniques for Marketing, Sales, and Customer Relationship Management (Second Edition)*, December 2008.
URL `http://www.data-miners.com/dmt_companion.htm`

[7] Burget, R.: Genetické algoritmy. In *Optimalizace*, 2007.
URL `http://www.vutbr.cz/elearning/mod/resource/view.php?id=53283`

[8] Han, J.; Kamber, M.: Data Mining: Concepts and Techniques. In *Data Mining: Concepts and Techniques (Second Edition)*, University of Illinois at Urbana-Champaign, 2003.

[9] Honzík, P.: Strojové učení. In *Skripta FEKT VUT v Brně*, 2006, str. 85.
URL `https://www.feec.vutbr.cz/et/skripta/uamt/Strojove_uceni_S.pdf?PHPSESSID=0f81a6ef518b222e650869b0444302cf`

[10] Janssens, D.; Wets, G.; Brijs, T.; aj.: Integrating Classification and Association Rules by proposing adaptations to the CBA Algorithm. In *Integrating Classification and Association Rules by proposing adaptations to the CBA Algorithm*, 2003.
URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.14.920`

[11] Jirsík, V.: Umělé neuronové sítě. In *Umělé neuronové sítě*, 2006.
URL http://jaja.kn.vutbr.cz/~belovic/MUIN/03_UMI_umele_neuronov%E9_s%EDt%EC.pdf

[12] Kotásek, P.: The data mining specification language = DMSL: Jazyk pro dolování z dat: short version of Ph.D. Thesis. In *DMSL: The data mining specification language = DMSL: Jazyk pro dolování z dat : short version of Ph.D. Thesis*, Brno, 2003, ISSN 80-214-2685-3, str. 27.

[13] Kárný, M.; Nedoma, P.; Šmídl, V.: Cross-validation of controlled dynamic models: Bayesian approach. In *Preprints of the 16th World Congress of the International Federation of Automatic Control*, 2005.
URL http://dar.site.cas.cz/?publication=312

[14] Lacko, L.: Databáze: datové sklady, OLAP a dolování dat s příklady v Microsoft SQL Serveru a Oracle. In *Databáze: datové sklady, analýza OLAP a dolování dat s příklady v SQL Serveru a Oracle*, Brno, 2003, ISBN 80-7226-969-0, str. 486.

[15] Li, W.; Han, J.; Pei, J.: CMAR: Accurate and Efficient Classification Based on Multiple Class-Association Rules. In *CMAR: Accurate and Efficient Classification Based on Multiple Class-Association Rules*, 2006.
URL http://www.cs.sfu.ca/~jpei/publications/cmar.pdf

[16] Norman, R.: Chapter 6: Decision Trees (Powerpoint slides). 2008.
URL http://www.data-miners.com/dmt_companion.htm

[17] Peng, W.; Chen, J.; Zhou, H.: An Implementation of ID3 - Decision Tree Learning Algorithm. In *An Implementation of ID3 - Decision Tree Learning Algorithm*, 2002.
URL http://web.arch.usyd.edu.au/~wpeng/DecisionTree2.pdf

[18] Pješivac-Grbović, J.; Fagg, G.; Angskun, T.; aj.: Decision Trees and MPI Collective Algorithm Selection Problem. In *Decision Trees and MPI Collective Algorithm Selection Problem*, 2006.
URL http://www.netlib.org/utk/people/JackDongarra/PAPERS/decision-tree-ipdps07.pdf

[19] Roach, C.: Building Decision Trees in Python. 2006.
URL http://www.onlamp.com/pub/a/python/2006/02/09/ai_decision_trees.html?CMP=OTC-UD6648202101&ATT=Building+Decision+Trees+in+Python

[20] Rud, O. P.: Data mining: praktický průvodce dolováním dat pro efektivní prodej, marketing a podporu zákazníků (CMR). In *Data Mining cookbook modeling data for marketing*, Praha, 2001, ISBN 80-7226-577-6, str. 329.

[21] Rychlý, M.: Klasifikace a predikce. In *Klasifikace a predikce*, 2006.
URL `http://www.fit.vutbr.cz/~rychly/docs/classification-and-prediction/classification-and-prediction.pdf`

[22] Senashova, M. Y.; Gorban, A. N.; Wunsch, D. C.: Back-propagation of accurancy. In *Back-propagation of accurancy*, 2003.
URL `http://arxiv.org/ftp/cond-mat/papers/0305/0305527.pdf`

[23] Žambochová, M.: Jak na rozhodovací stromy. In *Informační Bulletin České statistické společnosti*, 2008.
URL `http://www.statspol.cz/bulletiny/ib-2008-3.pdf`