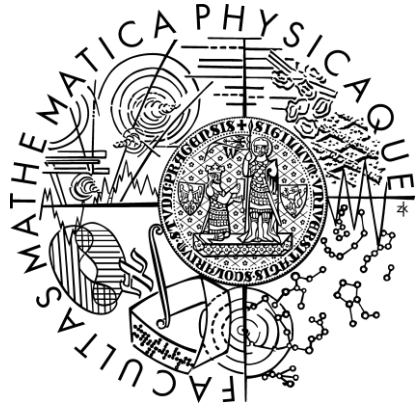
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

# DIPLOMOVÁ PRÁCE




Miroslav Jahoda

# Rozhodovací stromy

*Katedra teoretické informatiky a matematické logiky*
Vedoucí diplomové práce: *Doc. RNDr. Iveta Mrázová, CSc.*
Studijní program: *Informatika*
Studijní obor: *Teoretická informatika*

Praha 2009

Rád bych na tomto místě poděkoval všem, kteří mi pomáhali při vzniku této práce. Děkuji mé vedoucí diplomové práce Doc. RNDr. Ivetě Mrázové, CSc. za její cenné připomínky a trpělivost. Dále bych chtěl poděkovat celé mé rodině a obzvlášť mé snoubence Vendulce, bez jejichž podpory by tato práce nemohla vzniknout.

Prohlašuji, že jsem svou diplomovou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce.

V Praze dne 7.8.2009 Miroslav Jahoda

## Obsah

## Abstrakt

Název práce: *Rozhodovací stromy*
Autor: *Miroslav Jahoda*
Katedra (ústav): *Katedra teoretické informatiky a matematické logiky*
Vedoucí diplomové práce: *Doc. RNDr. Iveta Mrázová, CSc.*
e-mail vedoucího: *Iveta.Mrazova@mff.cuni.cz*
Abstrakt:

*Mezi známé metody dobývání znalostí patří neuronové sítě, ILP, asociační pravidla, Bayesovské sítě, klastrování, rozhodovací stromy a další. Tato práce se zabývá právě rozhodovacími stromy, jejich implementací, vizualizací, extrakcí pravidel a také porovnáváním různých rozhodovacích stromů a modelů pro klasifikaci dat vůbec. Nedílnou součástí procesu dobývání znalostí je také předzpracování dat, které hraje důležitou roli a je také rozebíráno v této práci. Součástí této práce je i porovnání různých modelů rozhodovacích stromů jako CART, CHAID, C5.0 (See5) a jiných na množině 3 druhů dat. Nakonec jsou výsledky porovnány s výsledky na předzpracovaných datech pomocí PCA analýzy.*

Klíčová slova: *Datamining, Rozhodovací stromy, Neuronové sítě, PCA, Předzpracování, Extrakce pravidel, Vizualizace*

Title: *Decision trees*
Author: *Miroslav Jahoda*
Department: *Department of Theoretical Computer Science and Mathematical Logic*
Supervisor: *Doc. RNDr. Iveta Mrázová, CSc.*
Supervisor's e-mail address: *Iveta.Mrazova@mff.cuni.cz*
Abstract:

*Among the known methods of data mining are neural networks, ILP, associative rules, Bayes networks, clustering, decision trees and others. This thesis is about decision trees, their implementation, visualization, extraction rules and the comparison of different decision trees and models of classification data in general. An integral part of the data mining process is preprocessing of data, which plays an important role and is also discussed in this thesis. Part of this thesis also concerns the comparison of different decision tree models such as CART, CHAID, C5.0 (See5) and others on a set of 3 data kinds. Finally, this model's results are compared to its results when data preprocessed by PCA analysis is used.*

Keywords: *Datamining, Decision trees, Neural Networks, PCA, Preprocessing, Extraction rules, Visualisation*

# 1 Úvod

Rozhodovací stromy patří do oboru nazývaném dobývání znalostí. Dobývání znalostí je ale široký pojem, pod který spadá mnoho různých metod. Cílem tohoto oboru je automaticky nalézat zajímavé informace z nashromážděných dat v ideálním případě bez zásahu člověka. Cílem rozhodovacích stromů je nalézat v datech skryté závislosti, které pomohou klasifikovat příklady do různých tříd (např. zda klient je dostatečně důvěryhodný, aby mu banka půjčila peníze). Nepřímým důsledkem tohoto procesu je možná jednoduchá extrakce pravidel, které slouží k dopočítání se k danému výsledku klasifikace.

Koncept rozhodovacích stromů je dobře znám nejen v oblasti výpočetní techniky. Používají ho například přírodovědci ve svých klíčích k určování rostlin a živočichů. Výhodou rozhodovacího stromu je jeho jednoduchá interpretace a možnost ho jednoduše převést na sadu rozhodovacích pravidel (indukcí). Rozhodovací stromy se dnes používají i například pro zefektivnění práce a zvýšení zisků popřípadě snížení ztrát.

Tato diplomová práce se věnuje rozhodovacím stromům od přípravy dat, přes vytváření až po následnou extrakci a interpretaci nalezených znalostí a jejich možných zobrazení a porovnání s jinými metodami, které měly zadány stejný cíl. Práce se věnuje také zlepšení schopností rozhodovacích stromů, kterého je možné dosáhnout i malými úpravami standardního způsobu vytváření rozhodovacích stromů a jejich úskalími.

Vstupní data pro rozhodovací stromy budou pro jednoduchost v celé práci myšlena jako tabulka, v které každý řádek představuje jeden příklad (např. nějakého klienta banky) a každý sloupec představuje jeden atribut (například velikost majetku klienta).

Rozhodovací stromy se vytvoří na základě předchozích zkušeností a dají se použít k odhadu klasifikace nových dat do daných tříd podle hodnot známých atributů. Rozhodovací stromy potřebují pro svoje vytvoření učitele (tj. už existující klasifikace podobných příkladů).

V závěru práce se srovnává několik různých systémů, které implementují rozhodovací stromy, na různých datech.

# 2 Základní model rozhodovacího stromu

K definici rozhodovacího stromu je potřeba znát nejdříve pár definic z teorie grafů:

### Definice 1: Strom

Strom je souvislý graf $T = (V, E)$ neobsahující kružnici. $V$ nebo také $V(T)$ je množina vrcholů a $E$ nebo také $E(T)$ je množina hran (tj. dvojic z množiny $V(T)$). Listy stromu $T$ jsou vrcholy z $V(T)$, které mají stupeň 1 (vede z nich jenom 1 hrana). Množina všech listových vrcholů stromu $T$ bude dále označována jako $V_L(T)$. Množina všech nelistových vrcholů (uzlů) pak $V_U(T)$.

### Definice 2: Kořenový strom

Kořenový strom je dvojice (T, r), kde T je strom a $r \in V(T)$ se nazývá kořen stromu T. Vrchol $v \in V(T)$ je otcem vrcholu $s \in V(T)$ pokud $\{v, s\} \in E(T)$ a $v$ leží na jediné cestě mezi kořenem $r$ a vrcholem $s$. Hranu s otcovským vrcholem $v$ a synovským vrcholem $s$ bude dále označována jako uspořádaná dvojice $(v, s) \in E(T)$.

Rozhodovací strom se pak zadefinuje následovně:

### Definice 3: Rozhodovací strom

Nechť $A$ je množina atributů (barva, datum, …) nebo otázek, $Hod(a \in A)$ je množina hodnot nebo odpovědí, které atribut $a$ nabývá a nechť $C$ je množina tříd, do kterých se příklady (data) mají zařadit na základě hodnot jednotlivých atributů z množiny $A$. **Rozhodovací strom** pro data s atributy $A$, hodnotami atributů $Hod(A)$ a třídami pro klasifikaci $C$, je uspořádaná čtveřice $\big((T, r), f_A, f_H, f_C\big)$, kde:

- $(T, r)$ je kořenový strom (viz teorie grafů) a $r \notin V_L(T)$.
- Zobrazení $f_A: V_U(T) \to A$ takové, že $\forall v \in V_U(T)\ \exists!\, a \in A: f_A(v) = a$ .
- Zobrazení $f_H: E(T) \to Hod(A)$ takové, že
$$\forall e \in E(T)\ \big(e = (v_1, v_2)\big)\ \exists h \in Hod\big(f_A(v_1)\big): f_H(e) = h$$
- Zobrazení $f_C: V_L(T) \to C$ takové, že $\forall l \in V_L(T)\ \exists c \in C: f_C(l) = c$.

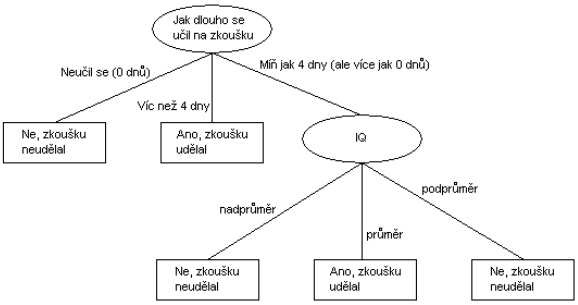
### Definice 4: Úplný rozhodovací strom

Úplný rozhodovací strom $\big((T, r), f_A, f_H, f_C\big)$ je rozhodovací strom, který splňuje navíc podmínku:
$$\forall v \in V_U(T)\ \forall h \in Hod\big(f_A(v)\big)\ \exists e \in E(T)\ \big(e = (v, v_2)\big): f_H(e) = h$$

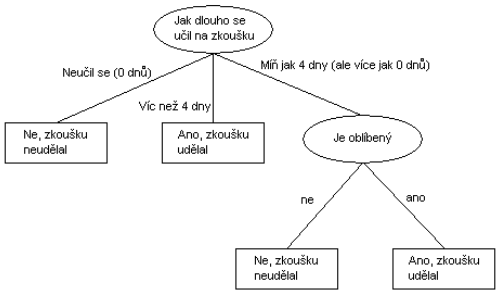
Pro lepší pochopení, jak rozhodovací stromy vypadají, jsou pro názornost v příkladu 1 vytvořeny dva různé úplné rozhodovací stromy (obrázek 1 a 2), klasifikující studenty do stejných tříd:

Příklad 1:

| Student | Je oblíbený | IQ | Jak dlouho se učil na zkoušku | Průměr známek | Zkoušku udělal(á) |
|---|---|---|---|---|---|
| Anna K. | Ano | Průměrné | Více než 4 dny | 1,3 | Ano |
| Milan H. | Ne | Průměrné | Více než 4 dny | 2,0 | Ano |
| Ladislav G. | Ne | Nadprůměrné | Méně jak 4 dny | 3,1 | Ne |
| Jan D. | Ne | Podprůměrné | Více než 4 dny | 3,2 | Ano |
| Jan T. | Ano | Podprůměrné | Méně jak 4 dny | 2,9 | Ne |
| Buhumír F. | Ano | Průměrné | Méně jak 4 dny | 2,5 | Ano |
| Libuše R. | Ano | Průměrné | Méně jak 4 dny | 3,0 | Ano |
| Rudolf J. | Ano | Průměrné | Neučil se | 1,4 | Ne |




Obrázek 1: Příklad rozhodovacího stromu pro příklad 1.

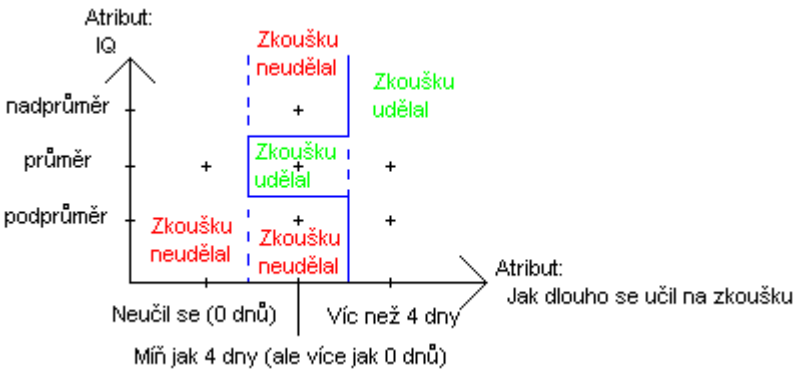



Obrázek 2: Příklad rozhodovacího stromu pro příklad 1.

Z výše uvedených příkladů plyne, že pro jednu sadu dat lze vytvořit různé rozhodovací stromy, které rozdělují data do stejných tříd.

Při vytváření rozhodovacího stromu z připravených dat se postupuje nejčastěji metodou rozděl a panuj. Data se postupně rozdělují na menší a menší skupiny, ve kterých převládají více a více příklady z jedné třídy, až nakonec zůstanou příklady čistě jenom z jedné třídy (listy). Tento postup se často označuje jako TDIDT (top down induction of decision trees) (viz kapitola 2.1). Rozhodovací stromy v podstatě postupně rozdělují prostor atributů nadrovinami rovnoběžnými s osami souřadné soustavy, což velmi zjednodušuje jejich interpretaci. Například rozdělení prostoru dat podle rozhodovacího stromu na obrázku 1 vypadá tak, jak zobrazuje obrázek 3.

Ne vždy je ale možné a žádoucí přesně klasifikovat trénovací data. Může to totiž vést k přeučení a tím k špatné klasifikaci dat, která v trénovací množině nemáme. Pro větší počet atributů a počet trénovacích dat může rozhodovací strom velmi narůst. Navíc při zašuměných trénovacích datech by výsledný rozhodovací strom stejně nebyl správný. Proto se přistupuje k prořezávání rozhodovacích stromů (viz kapitola 3.1) a požaduje se, aby v listovém uzlu převažovaly příklady z jedné ze tříd. Prořezaný strom bývá srozumitelnější pro interpretaci, míň náchylný k šumu a mnohem menší. Samozřejmě je to za cenu zhoršeného chování při klasifikaci trénovacích dat.




Obrázek 3: Rozdělení prostoru dat nadrovinami podle rozhodovacího stromu na obrázku 1.

Z příkladu 1 a z rozhodovacího stromu na obrázku 1 plyne, že výsledný rozhodovací strom nemusí popisovat správně logické rozdělení dat do jednotlivých tříd (logické by bylo, kdyby studentovi s nadprůměrným IQ stačila kratší příprava na zkoušku než studentovi s průměrným IQ). V uvedeném příkladu by pomohlo rozšíření trénovacích dat k vytvoření rozhodovacího stromu nebo rozšíření o další atributy, které obsahují informace o absolvování podobných předmětů nebo předmětů obsahujících alespoň část učiva, na kterou se může zkoušející na zkoušce zeptat. To klade důraz při vytváření rozhodovacích stromů nejen na samotné algoritmy, ale také na přípravu dat. V tomto konkrétním příkladu by se prořezáním rozhodovacího stromu moc nevyřešilo, protože by se skoro úplně ztratily příklady (řádky), kdy student zkoušku neudělá. Zůstala by jen varianta, kdy se student vůbec neučil. Což, až na pár výjimek, je hodně zkreslený pohled nevhodný k prezentování studentům k jejich inspiraci.

### 2.1 Algoritmus TDIDT

Algoritmus TDIDT je obecný algoritmus pro tvorbu rozhodovacích stromů. Algoritmus je následující:

**Algoritmus 1: TDIDT**

1. Nechť $D$ je množina dat (příkladů) s atributy $A$, které nabývají hodnot $Hod(A)$, pro která se má vytvořit rozhodovací strom. Zvolí se jeden atribut jako kořen podstromu. $f_A(r) = a \in A$.
2. Data $D$ se rozdělí v tomto uzlu na podmnožiny podle hodnot zvoleného atributu ($D_1$, $D_2$, … ). Přidá se uzel pro každou z podmnožin: $V = V \cup \{v_1, v_2, \dots\}$, $E = E \cup \{(r, v_1), (r, v_2), \dots\}$, $\forall v \in \{v_1, v_2, \dots\} \rightarrow f_H\big((r, v)\big) = h \in Hod\big(f_A(r)\big)$
3. Existuje-li uzel $v_i$, pro který nepatří všechna data $D_i$ do téže třídy $t \in T$, tak se spustí na $D_i$ algoritmus TDIDT a spojí se výsledný rozhodovací strom pro $D_i$ s právě vytvářeným tak, že $v_i = r_{D_i}$. Jinak pro každý vrchol $v_j$, pro který patří všechna data $D_j$ do téže třídy $t \in T$ přidej zobrazení $f_T\big(v_j\big) = t$.

Základní otázka tohoto algoritmu je, který z atributů se nejvíce hodí jako kořen daného podstromu. Je jasné, že cílem je vybrat atribut, který od sebe nejlépe rozlišuje dané třídy.

Teorie informace a pravděpodobnost poskytují charakteristiky atributů, které jsou k tomuto účelu nejvíce využívány. Jsou to např. entropie, informační zisk, poměrný informační zisk, Giniho index a $\chi^2$, které budou popsány v následujících kapitolách, nebo lze také použít vzdálenost mezi atributem a třídou a mnoho dalších.

### 2.1.1 Entropie

Entropie udává míru neuspořádanosti zkoumaného systému. Pro systém s konečným počtem možných stavů S = {$s_1$,$s_2$,…,$s_n$}, kde n < ∞ a pravděpodobnostními výskyty P($s_i$) (platí, že $\sum_{i=1}^{n} P(s_i) = 1$), je informační entropie definovaná jako:

$$\mathrm{H(S)} = -\sum_{\mathrm{i=1}}^{\mathrm{n}} \left(\mathrm{P(s_i)log_2\big(P(s_i)\big)}\right) \qquad (1)$$

kde je formálně zadefinováno: $0 \log_2 0 \equiv 0$

Entropie nabývá maximálních hodnot pro rovnoměrné diskrétní rozložení (tj. $P(s_i) = \frac{1}{n}\ \forall i \in \{1,2,\dots,n\}$ ) a minimálních hodnot pro absolutní nadvládu případů z jedné třídy (tj. $P(s_j) = 1\ \ kde\ j \in \{1,2,\dots,n\}\ \wedge\ P(s_i) = 0\ \ pro\ \forall i \neq j\ \wedge i \in \{1,2,\dots,n\}$ ).

Entropie $H(Hod(a) = h)$ hodnoty $h$ atributu $a$ vzhledem k třídám se spočítá podle vzorce entropie (1) na příkladech (řádcích), které v daném podstromu pokrývají kategorii $Hod(a) = h$ (tj. na všech příkladech v daném podstromu, jejichž atribut $a$ nabývá hodnoty $h$).

$$H(Hod(a) = h) = -\sum_{t=1}^{T} \left(\frac{n_t(Hod(a) = h)}{n(Hod(a) = h)} log_2 \left(\frac{n_t(Hod(a) = h)}{n(Hod(a) = h)}\right)\right) \qquad (2)$$

kde $T$ = počet tříd, do kterých se mají data zařadit,
$n_t(Hod(a) = h)$ = počet případů (řádků) v daném podstromu zařazených do $t$-té třídy jejichž atribut $a$ nabývá hodnoty $h$,
$n(Hod(a) = h)$ = počet případů (řádků) v daném podstromu, jejichž atribut $a$ nabývá hodnoty $h$.

Entropie $H(a)$ pro jeden atribut $a$ se pak vypočítá jako vážený součet entropií každé hodnoty, které může atribut nabývat:

$$H(a) = \sum_{h \in \mathrm{Hod(a)}} \left(\frac{n(Hod(a) = h)}{n} H(Hod(a) = h)\right) \qquad (3)$$

kde $Hod(a)$ je množina všech hodnot, které může nabývat atribut $a$,
$n(Hod(a) = h)$ je počet případů (řádků) v daném podstromu, jejichž atribut $a$ nabývá hodnoty $h$,
$n$ je počet všech případů (řádků) v daném podstromu,
$H(Hod(a) = h)$ je entropie hodnoty $h$ atributu $a$ v daném podstromu.

Pro větvení stromu se pak vybere atribut s nejmenší entropií $H(a)$.
Například entropie atributu „Jak dlouho se učil student na zkoušku“ ($a$) v příkladu 1 se spočítá následovně:

| **Jak dlouho se učil na zkoušku** | **Počet studentů, kteří zkoušku udělaly** | **Počet studentů, kteří zkoušku neudělaly** | $H(Hod(a) = h)$ |
|---|---|---|---|
| Neučil se | 0 | 1 | $-\left(\left(\frac{0}{1}\log_2\frac{0}{1}\right)+\left(\frac{1}{1}\log_2\frac{1}{1}\right)\right)=0$ |
| Méně jak 4 dny | 2 | 2 | $-\left(\left(\frac{2}{4}\log_2\frac{2}{4}\right)+\left(\frac{1}{4}\log_2\frac{1}{4}\right)\right)=1$ |
| Více jak 4 dny | 3 | 0 | $-\left(\left(\frac{3}{3}\log_2\frac{3}{3}\right)+\left(\frac{0}{3}\log_2\frac{0}{3}\right)\right)=0$ |

$$H(\text{"}Jak\ dlouho\ se\ učil\ na\ zkoušku\text{"}) = \left(\left(\frac{1}{8}0\right)+\left(\frac{4}{8}1\right)+\left(\frac{3}{8}0\right)\right) = 0{,}5 \qquad (4)$$

### 2.1.2 Informační zisk

Informační zisk i poměrný informační zisk jsou míry odvozené z entropie. Informační zisk měří redukci entropie způsobenou volbou atributu $a$. Spočítá se jako:

$$Zisk(a) = H(A) - H(a) \qquad (5)$$

kde $H(A)$ je entropie celku (všech dat v daném podstromu) vzhledem na cílové třídy

$H(A) = -\sum_{t=1}^{T}\left(\frac{n_t}{n}\log_2\frac{n_t}{n}\right)$

$T$ = počet tříd, do kterých se mají vstupní data zařadit

$n_t$ =počet případů (řádků) odpovídající $t$-té třídě v trénovacích datech

$n$ = počet všech případů (řádků) v trénovacích datech

$H(A)$ je nezávislá na atributu $a$ tedy pro výpočet všech entropií jednotlivých atributů v rámci hledání kořene podstromu se tedy chová jako konstanta. Takže pokud se v případě entropie hledá atribut s nejmenší entropií, tak v informačním zisku to odpovídá hledání maximálního zisku.
Například informační zisk atributu „Jak dlouho se učil student na zkoušku“ ($a$) v příkladu 1 se spočítá následovně:

$$H(A) = -\left(\left(\frac{5}{8}\log_2\frac{5}{8}\right)+\left(\frac{3}{8}\log_2\frac{3}{8}\right)\right) \cong 0{,}954434$$

$$H(\text{"Jak dlouho se učil na zkoušku"}) = 0{,}5 \qquad \text{viz vzorec (4)}$$

$$Zisk(\text{"}Jak\ dlouho\ se\ učil\ na\ zkoušku\text{"}) \cong 0{,}954434 - 0{,}5 = 0{,}454434 \qquad \text{viz (5)}$$

Poměrný informační zisk, na rozdíl od entropie a informačního zisku, bere v úvahu také počet hodnot jednotlivých atributů:

$$Poměrný\ zisk(a) = \frac{Zisk(a)}{Větvení(a)} \qquad (6)$$

kde *Větvení(a)* je vlastně entropie dat vzhledem k hodnotám atributu $a$ (ne třídám):

$$Větvení(a) = -\sum_{\mathrm{h}\in Hod(a)} \left( \frac{n(Hod(a)=h)}{n} log_2 \frac{n(Hod(a)=h)}{n} \right) \quad (7)$$

kde $n(Hod(a)=h)$ je počet příkladů (řádků) v daném podstromu, kde atribut $a$ nabývá hodnoty $h$,

$n$ je celkový počet příkladů (řádků) v daném podstromu

Poměrný informační zisk nám odstraňuje nevýhodu slepého pohledu, jak moc dobře daný atribut odlišuje příklady různých tříd. Pokud by například byl jeden z atributů pořadové číslo, tak by měl nejmenší entropii, ale pro klasifikaci nových dat by byl zcela nepoužitelný (klasifikoval by jenom trénovací data).

Například poměrný informační zisk atributu „Jak dlouho se učil student na zkoušku“ ($a$) v příkladu 1 se spočítá následovně:

$$Zisk("Jak\ dlouho\ se\ učil\ na\ zkoušku") \cong 0{,}454434 \quad \text{viz vzorec (5)}$$

$$Větvení(A) = -\left(\left(\frac{1}{8}\log_2\frac{1}{8}\right) + \left(\frac{3}{8}\log_2\frac{3}{8}\right) + \left(\frac{4}{8}\log_2\frac{4}{8}\right)\right) \cong 1{,}405639$$

$$Poměrný\ zisk("Jak\ dlouho\ se\ učil\ na\ zkoušku") \cong \frac{0{,}454434}{1{,}405639} \cong 0{,}3232935 \quad (8)$$

### 2.1.3 Giniho index

Podobnou roli jako entropie může poskytnout také Giniho index. Tento index dává do souvislosti obsah plochy pod ideální křivkou a skutečnou křivkou a dá se jednoduše vyjádřit jako:

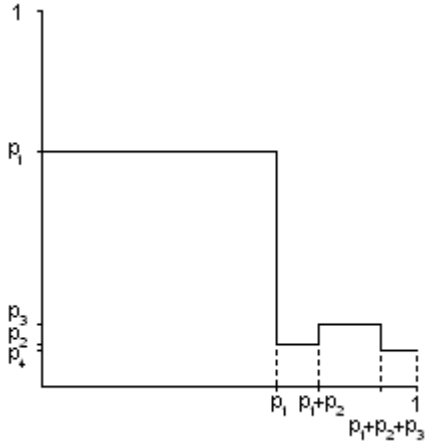
$$Gini = \frac{ObsahIdeálníPlochy - ObsahSkutečnéPlochy}{ObsahIdeálníPlochy} \quad (9)$$

Pokud se zadefinuje křivka rozdělení příkladů (řádků) do tříd takto:

$$y = p_t \quad pro\ x \in \left\langle \sum_{i=1}^{t-1} p_i , \sum_{i=1}^{t} p_i \right) \ \wedge \ \ t \in \{1,2,\dots,T\} \quad (10)$$

kde $T$ = počet tříd, do kterých se mají data zařadit

$p_i$ = pravděpodobnost, že příklad (řádek) bude zařazen do třídy $\boldsymbol{i}$ a platí $\sum_{t=1}^{T} p_t = 1$.




Obrázek 4:Křivka pravděpodobností zařazení jednotlivých příkladů do tříd.

tak obsah pod touto křivkou bude roven $\sum_{t=1}^{T}(p_t^2)$.

Ideální křivka by v tomto případě byla pro případ, kdy s pravděpodobností 1 budou všechny příklady (řádky) zařazeny do jedné a té samé třídy (tj. pravděpodobnosti zařazení příkladů do ostatních tříd bude rovno 0).
Giniho index bude tedy vypadat takto:

$$Gini = \frac{1^2 - \sum_{t=1}^{T}(p_t^2)}{1^2} = 1 - \sum_{t=1}^{T}(p_t^2) \tag{11}$$

kde $T$ je počet tříd
$p_t$ je pravděpodobnost, že příklady (řádky) z daného podstromu spadají do $t$-té třídy. Tato pravděpodobnost se odhaduje jako:

$p_t = \frac{n_t}{n}$ kde $n$=počet všech příkladů z trénovacích dat a
$n_t$=počet příkladů spadajících do $t$-té třídy

Giniho index pro jeden atribut se spočítá analogicky, jako u entropie (vzorce [2] a [3]):

$$Gini(Hod(a) = h) = 1 - \sum_{t=1}^{T}\left(\frac{n_t(Hod(a) = h)}{n(Hod(a) = h)}\right)^2 \tag{12}$$

kde $T$ je počet tříd, do kterých se mají data zařadit,
$n_t(Hod(a) = h)$ je počet případů (řádků) v daném podstromu zařazených do $t$-té třídy jejichž atribut $a$ nabývá hodnoty $h$,
$n(Hod(a) = h)$ je počet případů (řádků) v daném podstromu, jejichž atribut $a$ nabývá hodnoty $h$.

$$Gini(a) = \sum_{h \in Hod(a)}\left(\frac{n(Hod(a) = h)}{n} Gini(Hod(a) = h)\right) \tag{13}$$

kde $n(Hod(a) = h)$ je počet případů (řádků) v daném podstromu, jejichž atribut $a$ nabývá hodnoty $h$.
$n$ je celkový počet příkladů (řádků) v daném podstromu

Atribut s nejmenším Giniho indexem se pak použije jako další kořen podstromu v rozhodovacím stromu.
Například Giniho index pro atribut „Jak dlouho se učil student na zkoušku“ ($a$) v příkladu 1 se spočítá následovně:

$Gini(Hod(a) = \text{"}Neučil\ se\text{"}) = 1 - \left(\left(\frac{0}{1}\right)^2 + \left(\frac{1}{1}\right)^2\right) = 0$ viz vzorec (12)

$Gini(Hod(a) = \text{"}Míň\ jak\ 4\ dny\text{"}) = 1 - \left(\left(\frac{2}{4}\right)^2 + \left(\frac{2}{4}\right)^2\right) = \frac{1}{2}$ viz vzorec (12)

$Gini(Hod(a) = \text{"}Víc\ než\ 4\ dny\text{"}) = 1 - \left(\left(\frac{3}{3}\right)^2 + \left(\frac{0}{3}\right)^2\right) = 0$ viz vzorec (12)

$Gini(\text{"Jak dlouho se učil na zkoušku"}) = \left(\left(\frac{1}{8} \times 0\right) + \left(\frac{4}{8} \times \frac{1}{2}\right) + \left(\frac{3}{8} \times 0\right)\right) = 0{,}25$ viz vzorec (13)

Analogicky s informačním ziskem se může také maximalizovat rozdíl $\left(Gini(A) - Gini(a)\right)$
kde $Gini(A) = 1 - \sum_{t=1}^{T}\left(\frac{n_t}{n}\right)^2$

### 2.1.4 $\chi^2$

Statistikou $\chi^2$ se zjišťuje, jak moc jsou na sobě nezávislé nějaké veličiny. V našem případě si můžeme toto trochu přeformulovat a ptát se, jak moc je platná hypotéza, že daný atribut je nezávislý na dané třídě. Tím dostáváme požadované hodnocení, jak moc je vhodné vybrat daný atribut k rozdělení požadovaných dat.

Mějme kontingenční tabulku (viz Tabulka 1), v které $t_1$ až $t_S$ jsou všechny třídy, do kterých mohou být data klasifikována a $h_1$ až $h_R$ jsou hodnoty, kterých může vybraný atribut nabývat. $a_{ij}$ jsou pak četnosti, které říkají, kolik příkladů nabývajících v daném atributu hodnoty $h_i$ spadá do třídy $t_j$.

| | $t_1$ | $t_2$ | … | $t_S$ | $\Sigma$ |
|---|---|---|---|---|---|
| $h_1$ | $a_{11}$ | $a_{12}$ | … | $a_{1S}$ | $r_1$ |
| $h_2$ | $a_{21}$ | $a_{22}$ | … | $a_{2S}$ | $r_2$ |
| ⋮ | ⋮ | ⋮ | … | ⋮ | ⋮ |
| $h_R$ | $a_{R1}$ | $a_{R2}$ | … | $a_{RS}$ | $r_R$ |
| $\Sigma$ | $s_1$ | $s_2$ | … | $s_S$ | $n$ |

Tabulka 1: Kontingenční tabulka vybraného atributu a jednotlivých tříd

Pak statistika $\chi^2$ se spočítá jako:

$$\chi^2 = \sum_{i=1}^{R}\sum_{j=1}^{S}\frac{\left(a_{ij} - \frac{r_i s_j}{n}\right)^2}{\frac{r_i s_j}{n}} \tag{14}$$

Existuje $\chi^2$ test, který pro vypočítanou statistiku $\chi^2$ s daným stupněm volnosti (R-1)(S-1) říká, zda je hypotéza o nezávislosti platná, či nikoli pro danou hladinu významnosti $\alpha$. Pokud tedy:

$$\chi^2 \geq \chi^2_{(R-1)(S-1)}(\alpha)\,, \tag{15}$$

tak se zamítne hypotéza o nezávislosti. Tento test je ale asymptotický a měl by se provádět jenom pro dostatečně velký počet příkladů (n). Uvádí se, že musí platit:

$$\frac{r_i s_j}{n} \geq 5 \;\; pro\; \forall i = 1, \dots, R \;\; \forall j = 1, .. S \tag{16}$$

V poslední době se však spíše používá Yarnoldovo kritérium (viz (Anděl, 2002)):

$$\frac{r_i s_j}{n} \geq 5q \;\; pro\; \forall i = 1, \dots, R \;\; \forall j = 1, .. S, \tag{17}$$

kde $q$ je podíl tříd, pro něž platí $\frac{r_i s_j}{n} < 5$.

Pokud se hypotézu o nezávislosti nepodaří podle tohoto testu prokázat pro atribut, který vyšel jako kandidát pro uzel (tj. s největší spočtenou statistikou $\chi^2$), tak nemá dále cenu vytvářet rozhodovací strom tímto směrem. Pokud ovšem nebude splněna podmínka (16), tak je test neprůkazný a bude se v tvorbě stromu pokračovat. V případě, že pro daná data není

klasifikován ani jeden příklad do nějaké třídy, tak tuto třídu do kontingenční tabulky vůbec nezahrneme. Tím zvětšíme pravděpodobnost legitimního provedení testu a odpověď, jestli je atribut nezávislý na dané třídě, či nikoli.

# 3 Vylepšení rozhodovacích stromů

## 3.1 Prořezávání rozhodovacích stromů

Rozhodovací stromy vytvářené pomocí algoritmu TDIDT trpí velmi přeučením. Přeučení je stav, kdy rozhodovací strom funguje dobře na testovacích datech, ale na dalších datech, která nebyla použita pro vytváření rozhodovacího stromu, se chová naprosto špatně. Vyplývá to z omezeného množství dat pro testování, která nemusejí obsahovat všechny možné varianty, anebo obsahují zašuměná data. Jedním ze způsobů, jak se lze vyhnout přeučení, je prořezání rozhodovacího stromu.

Existují 2 základní varianty prořezávání:

1. Prořezávání již při vytváření rozhodovacího stromu.
2. Prořezávání už vytvořeného rozhodovacího stromu.

První možnost se implementuje jako modifikace algoritmu vytváření. Druhá možnost se implementuje jako další zpracování stromu, což dává více možností a algoritmus pro prořezávání má k dispozici více informací. Proto se v praxi používá více 2. možnost. Dále bude proto podrobněji popsána pouze tato možnost.

Pro prořezávání hotového rozhodovacího stromu se postupuje dle algoritmu 2:

### Algoritmus 2: Prořezávaní úplného rozhodovacího stromu

1. Seřaď nelistové uzly podle vzdálenosti od kořene (čím větší vzdálenost, tím dřív ke zpracování) a postupně pro ně proveď kroky 2 a 3.
2. Spočítej pro vybraný uzel správnost (viz dále) pro varianty, že
   - se uzel nezmění,
   - se uzel nahradí listem (třídou) s největším počtem zařazených příkladů v tomto podstromu,
   - se vybraný uzel nahradí synovským uzlem (v praxi se porovnává jenom synovský uzel s největším zastoupením příkladů v daném podstromu).
3. Pro variantu s největší správností proveď požadovanou akci.

Největší problém tohoto algoritmu je, jak lze spočítat správnost dané varianty. K tomuto účelu se používá statistický test na testovacích nebo na validačních datech (data nepoužitá při vytváření rozhodovacího stromu).

Správnost dané varianty se dá vyjádřit jako pesimistický odhad pro počet špatně zařazených příkladů daným rozhodovacím stromem.

V případě rozhodovacích stromů máme k dispozici informaci, kolik příkladů se špatně klasifikuje (nechť je to $F$) z daného počtu příkladů (nechť je to $n$). Nechť $y_i \sim Bi(1, q)$ tj. $y_i$ nabývá hodnot 0 a 1 a má binomické rozdělení. V tomto případě nechť $y_i$ vyjadřuje, zda i-tý příklad byl klasifikován rozhodovacím stromem špatně $(y_i = 1)$ a tedy $\sum_i y_i = F$. Střední hodnota $Ey_i = q$ a rozptyl $\sigma^2 = q(1 - q)$.

Jeden z možných statistických testů může být určení mezní hodnoty $F^*$, pro kterou platí, že s pravděpodobností $1 - \frac{\beta}{2}$ bude počet špatně klasifikovaných příkladů menší než $F^*$.

Pro určení hodnoty $F^*$ se využije Centrální limitní věta (viz (Anděl, 2002)), která platí pro $n \to \infty$ (konečný rozptyl, …) a říká:

$$\frac{\sum_i y_i - nq}{\sqrt{n}} \xrightarrow{d} N\big(0, q(1-q)\big) \tag{18}$$

kde $N\big(0, q(1-q)\big)$ je normální rozdělení se střední hodnotou 0 a rozptylem $q(1-q)$

z čehož potom plyne:

$$X = \frac{\frac{1}{n}}{\frac{1}{n}} \frac{\sum_i y_i - nq}{\sqrt{n}} \div \sqrt{q(1-q)} = \frac{\frac{\sum_i y_i}{n} - q}{\sqrt{\frac{q(1-q)}{n}}} \xrightarrow{d} N(0,1) \tag{19}$$

Po malých úpravách se pak získá:

$$-z \le X \le z \;\rightarrow\; -z \le \frac{\frac{\sum_i y_i}{n} - q}{\sqrt{\frac{q(1-q)}{n}}} \le z \;\rightarrow\; -z \le \frac{\frac{F}{n} - q}{\sqrt{\frac{q(1-q)}{n}}} \le z \tag{20}$$

kde $P(-z \le X \le z) = 1 - \beta$.

Pro výpočet $z$ se dají použít ve statistice tabulky, ve kterých jsou spočítané hodnoty z pro dané pravděpodobnosti. Jenom pro úplnost, statistické tabulky kvantilů jsou uváděné pro $P(X < u_\alpha) = \alpha$, ale u normálního rozdělení platí, že $P(-z \le X \le z) = 1 - 2\,P(X > z) = 1 - 2\,P(X < -z)$. (Výpočet kvantilu pro danou pravděpodobnost dokáže spočítat i MS Excel: $NORMINV(\alpha; 0; 1) = u_\alpha$.)

| $P(-z \le X \le z)$ | $P(X < -z)$ | $z$ |
|---|---|---|
| 99% | 0,5% | 2,575829 |
| 90% | 5% | 1,644854 |
| 80% | 10% | 1,281552 |
| 70% | 15% | 1,036433 |
| 60% | 20% | 0,841621 |
| 50% | 25% | 0,67449 |
| 40% | 30% | 0,524401 |
| 30% | 35% | 0,38532 |
| 20% | 40% | 0,253347 |
| 10% | 45% | 0,125661 |

Tabulka 2: Kvantily pro normální rozdělení $N(0,1)$

Protože jedinou neznámou ve vzorci (20) je $q$ a to vyjadřuje pravděpodobnost špatně zařazených příkladů, tak pro pesimistický odhad budeme chtít $q$ maximalizovat (tj. větší pravděpodobnost chybného zařazení příkladů).

Nerovnice (20) dává následující možnosti:

1. možnost: $\left(q \leq \frac{F}{n}\right) \wedge (q \in \langle q_1, q_2 \rangle)$
2. možnost: $\left(q \geq \frac{F}{n}\right) \wedge (q \in \langle q_1, q_2 \rangle)$

$$kde \quad q_{1,2} = \frac{2F + z^2 \pm z\sqrt{4F - 4\frac{F^2}{n} + z^2}}{2(n + z^2)}$$

Vzhledem k tomu, že platí $\frac{F}{n} \leq q_2$ , tak pesimistická pravděpodobnost bude vždy rovna $q_2$.

$q_2$ se dá tedy použít jako kvalifikátor, kterým se dají dva podstromy porovnat vzhledem k pesimistickému odhadu schopnosti daného podstromu dobře klasifikovat data (hledá se co nejmenší $q_2$). Tímto způsobem se dají porovnávat také dva různé rozhodovací stromy. Tento kvalifikátor má ale tu nevýhodu, že není dostatečně přesný pro malé n (vzhledem k tomu, že se použila věta, která má za předpoklad $n \to \infty$.

### 3.2 Ohodnocování atributů cenami

Někdy je pro získání hodnoty nějakého atributu zapotřebí většího úsilí než na získání hodnot jiných atributů. V tom případě je dobré ohodnotit atributy cenami, které budou hrát při vytváření rozhodovacího stromu také svoji úlohu. Příkladem ceny může být nejenom cena v penězích, ale také čas nebo i jak moc může odpověď diskreditovat či už nějakou společnost nebo jedince.

Zahrnutí ceny do rozhodovaní při vytváření rozhodovacího stromu může být například pro informační zisk následující:

$$\frac{Zisk(A)^2}{Cena(A)} \tag{21}$$

# 4 Předzpracování dat

Předzpracování dat má při dobývání znalostí velký vliv na výsledek. Někdy je dokonce i nutné, aby mohly vůbec nějaké výsledky být. Vždy záleží na tom, jakou techniku pro dobývání použijeme. V případě rozhodovacích stromů je například důležité, aby měly atributy omezený počet hodnot (tj. nejsou vhodné atributy s oborem hodnot reálných čísel). Také není vhodné mít hodně atributů, protože se čas pro vytvoření rozhodovacího stromu rapidně zvyšuje. Velkým problémem jsou dále strukturovaná data. Například 2 chemické sloučeniny složené ze stejného počtu daných atomů ale s jiným uspořádáním (stačí i jenom spinem) mohou vykazovat jiné vlastnosti ve své účinnosti. Možnosti, jak tyto problémy řešit, jsou popsány v následujících kapitolách.

## 4.1 Atributy s příliš mnoha hodnotami

Atributy, které nabývají příliš mnoha hodnot, nejsou pro vytváření rozhodovacího stromu velmi vhodné, jelikož jsou s nimi spojeny následující problémy:

1. Způsobují velké větvení stromu.
2. Tímto větvením mohou způsobovat mnohem menší schopnost rozhodovacího stromu správně zobecňovat.
3. Způsobují větší časovou náročnost při rozhodování, zda daný atribut má být vybrán do dalšího rozhodovacího uzlu.

U těchto atributů se provádí seskupování hodnot do skupin. V každé skupině by měli být co nejpodobnější hodnoty s ohledem na výslednou klasifikaci. To může být provedeno například pomocí postupného slučování hodnot a následnou kontrolou pomocí jedné z metod výpočtu vhodnosti jako např. Giniho index a jiné (viz podkapitoly v kapitole 2.1). Specifický příklad atributů s mnoha hodnotami jsou numerické atributy, které jsou podrobněji popsány v následující podkapitole.

### 4.1.1 Numerické atributy

Numerické atributy nabývají velkého počtu hodnot. Vytvoření samostatných větví pro každou nabytou hodnotu je nesmyslné už jen proto, že s velkou pravděpodobností (hlavně v případě, kdy mohou hodnoty být z nějakého spojitého intervalu reálných čísel) bude jednu hodnotu nabývat jenom jeden příklad z dat. To způsobí velké rozvětvění uzlů při vytváření stromu. Jakmile ale přijdou další příklady, které v trénovacích datech nebyly, tak tyto nové příklady nebude strom schopný správně klasifikovat.

U numerických atributů platí (až na malý počet výjimek), že čím jsou dvě hodnoty blíže u sebe, tím spíše budou mít podobné vlastnosti a vliv na případné zařazování dat do daných tříd. Proto se rozděluje obor hodnot numerického atributu do intervalů a ty pak slouží jako hodnoty (lze si to představit, jako nahrazení numerického atributu umělým atributem, který bude mít za hodnoty intervaly možných hodnot původního atributu).

Způsobů (algoritmů), jak umělý atribut vytvořit, existuje více. Některé vytváří přesně daný počet intervalů a jiné je zase vytvářejí rozdělováním či slučováním již existujících intervalů. Nejznámější algoritmy jsou Algoritmus Fayyada a Iraniho nebo Leeho a Shinův algoritmus, ale i další, které se dají nalézt například v (Berka, 2003) nebo na internetu.

Ne vždy je ale vhodné provádět nahrazování numerických atributů za intervalové před procesem vytváření rozhodovacího stromu. Většinou je vhodnější provádět nahrazování v 2. kroku algoritmu TDIDT (viz Algoritmus 1). V nejjednodušším případě se provádí rozdělení jenom na dva intervaly, kdy se používá stejné kritérium jako při rozhodování, který atribut zvolit pro další rozhodovací uzel. Jediný rozdíl je, že se místo atributů procházejí dělící body, které rozdělí obor hodnot numerického atributu na dva intervaly. Rozdělením oboru hodnot pouze na dva intervaly se sice zvětšuje hloubka stromu, ale zároveň dochází k větší flexibilitě při jeho vytváření. Snížení hloubky stromu se pak může provádět zpětným průchodem, kdy se hledají 2 uzly (jeden uzel je synem toho druhého), které reprezentují stejný atribut dat (pseudo-atributy vytvořené s jednoho a toho samého původního numerického atributu). Tyto 2 uzly se spojí do jednoho a vytvoří se odpovídající intervaly.

Algoritmus Fayyada a Iraniho zobecňuje popsaný postup vytváření dvou intervalů hodnot. Tento algoritmus patří asi mezi nejznámější. Jedná se o vytváření intervalů shora dolů, tedy o postupné rozdělování už existujících intervalů. Algoritmus rekurzivně rozdělí interval jedním dělícím bodem (tj. rozdělí interval na dva podintervaly). Pro rozhodování používá informačního zisku. Tento algoritmus má navíc podmínku, zda se daný interval má rozdělit, která vychází z kritéria minimální deskripčí délky. Tato podmínka se dá navíc použít také v případě, kdy informační zisk bude pro 2 nebo více dělících bodů stejný.

### Algoritmus 3: Fayyadův a Iraniho algoritmus

1. Data se uspořádají vzestupně podle hodnoty daného numerického atributu.
2. Nalezne se nejvhodnější dělící bod $D$ na daném intervalu $Int$ a určí se pro něj $Zisk(A_{Int,D})$.
3. Pokud je $Zisk(A_{Int,D}) > \frac{\log_2(n-1)}{n} + \frac{\Delta A_{Int,D}}{n}$

   $\Delta A_{Int,D} = \log_2(3^k - 2) - kH(A_{Int}) - k_1 H(A_{Int_{\leq D}}) - k_2 H(A_{Int_{>D}})$

   kde $k =$ počet různých tříd, do kterých spadají příklady z intervalu $Int$

   $k_1 =$ počet různých tříd, do kterých spadají příklady z intervalu $Int_{\leq D}$

   $k_2 =$ počet různých tříd, do kterých spadají příklady z intervalu $Int_{>D}$

   $H(A_{interval}) =$ entropie dat vzhledem k třídám klasifikace na daném intervalu

   3.1. Rozděl interval Int na intervaly $Int_{\leq D}$ a $Int_{>D}$

   3.2. Pro intervaly $Int_{\leq D}$ a $Int_{>D}$ spusť znova tento algoritmus a tím je rozděl.
4. Pokud podmínka v kroku 3 nebyla splněna, tak skonči.

Samozřejmě se tento algoritmus dá upravit, pokud se preferuje jiný způsob výpočtu vhodnosti dělícího bodu. Například by se informační zisk dal nahradit $\chi^2$ a kritérium ukončení dělení intervalu $\chi^2$ testem.

## 4.2 Nestrukturovaná data

Nestrukturovaná data patří mezi ty nejhůře uchopitelné. Pro každý typ těchto dat se musí postupovat individuálně. Nejznámější data tohoto typu jsou texty, kde hraje významnou roli kontext, který se dá velmi těžce nahradit. Jeden přístup ke zpracování textů je ten, že se vytvoří atributy, které odpovídají slovům ze slovníku a přiřadí se do nich četnosti jednotlivých slov v textu případně jenom v odstavcích. Tento přístup má nevýhodu velkého počtu většinou málo využitých atributů a absence kontextu. Kontext se neobratně obchází pomocí frází, které se přidávají jako nové atributy. V tomto směru se v současné době vyvíjí velké úsilí, protože jenom malý krůček od textu jsou webové stránky a trošku větší krok multimédia (obrázky, audio a video, …).

## 4.3 Strukturovaná data

Velkým problémem pro automatické zpracování dat je také to, že ne jenom samotné hodnoty atributů mají vliv na výslednou klasifikaci, ale také možné závislosti mezi atributy, které se jednou hodnotou daného atributu nedají vystihnout. Strukturovaná data jsou různého druhu a ke každým datům se musí přistupovat jinak. Snad nejrozšířenější typy strukturovaných dat jsou:

1) časová nebo sekvenční data
2) prostorová data
3) strukturální data

### 4.3.1 Časová data

Časová data jako například sled opakovaných vyšetření, vývoj kurzů akcií, … nebo sekvenční data, kdy postupná měření nemusí mít jenom časové odůvodnění, jsou asi nejčastěji se vyskytující typ dat. Cílem pro jejich zpracování je určit trend, sezónní chování, cykly a šum. Tyto čtyři parametry mohou ve většině případů postačovat k porovnání různých časových dat stejného typu. U časových dat se často objevuje periodické chování, které může zjednodušit jejich popis. K analýze takových časových dat, jako jsou signály nebo jiná periodická data, se s velkou účinností používá Fourierova analýza jejímž výsledkem je spektrum daného signálu a umožňuje získat jeho časově-frekvenční popis (viz (Louis, a další, 1997)). Tato metoda je založena na okně, které v čase ohraničuje krátký úsek signálu a umožňuje tak určovat spektrum signálu jenom daného časového úseku. Lze dosáhnout optimálního poměru rozlišitelnosti v čase a frekvenci vhodnou změnou šířky okna v čase a ve tvaru. Další způsob nalezení periodičnosti v datech nabízí autokorelace. Trend se dá také určit například pomocí PCA analýzy (viz kapitola 4.5.2). Dalšími statistickými údaji jako je maximum, minimum, směrodatná odchylka a jinými se dá pak tento trend, nejčastěji zapsaný formou vektorů, zpřesnit.

### 4.3.2 Prostorová data

Prostorová data (například informace z GIS – geografický informační systém) nehrají důležitou roli jenom při zobrazování nalezených znalostí, ale také při klasifikaci. Mohou

nepřímo také přidávat další informace. Například při výzkumu, zda je nějaký živočich s nějakými vlastnostmi schopen přežít na zvoleném území určitý čas, hraje důležitou roli nejenom jaká potrava je na daném místě k dispozici, ale také s jakými dalšími územími je v sousedství a jaký potenciální nepřátelé mohou z těchto území přijít. Z tohoto důvodu se nejčastěji bere relace sousedství daných oblastí (území) a vytváří se agregované atributy (viz kapitola 4.4).

### 4.3.3 Strukturální data

Strukturální data, jako jsou například chemické sloučeniny, jsou na popis atributy asi nejsložitější, pokud se nemá použít variabilního počtu atributů. Samozřejmě chemický vzorec se dá napsat jako jeden řetězec. Ale:

1. Pro tvorbu rozhodovacích stromů je tento řetězec chápán v podstatě jako celek a nezkoumá se proč je řetězec takový jaký je.
2. Ani chemický vzorec neobsahuje všechny informace o dané chemické sloučenině. Vyjadřuje jenom vázanost atomů nebo jejich skupin na další atomy a nevyjadřuje už například spin.

Bohužel při tvorbě rozhodovacích stromů se pro tato data dají použít jenom statistické hodnoty jako například pro chemické sloučeniny počty jednotlivých atomů nebo skupin, počet a typy vazeb a jiné. Každý typ strukturálních dat je jiný a pro každý z nich se vytvářejí jiné typy statistických údajů. Pro klasifikaci dat, u kterých klasifikace může velmi záviset na dané struktuře, se používají spíše jiné metody získávání znalostí než rozhodovací stromy.

## 4.4 Data z multirelační databáze

Ideálně zadaná data pro vytvoření rozhodovacího stromu jsou uložena ve formě jedné tabulky, kde sloupce představují atributy a řádky jednotlivé příklady. Ne vždy jsou ale tato data zadána formou jedné tabulky. Častokrát bývají data uložena v multirelační databázi. Ta bohužel není vhodná jako vstup pro tvorbu rozhodovacího stromu a je nutné z ní vytvořit jednu tabulku.

Při vytváření jedné tabulky je nutné dávat pozor, jaká data a jaké atributy v ní nakonec budou. Relace s kardinalitou 1:1 je jednoduchá a dochází pouze ke spojení dvou tabulek do jedné.

Relace s kardinalitou 1:n je už složitější. Některé relace mohou být časového původu, a tedy se s nimi musí zacházet podle toho (viz kapitola 4.3). Jiné relace mohou vyjadřovat například vlastnictví (klient má více účtů). Obecně je nejlepší, pokud vytvoření jedné vstupní tabulky provede expert, který má dobré povědomí o vlastnostech jednotlivých relací. Pokud expert není k dispozici, tak se provádí jednoduchá agregace hodnot (pro numerické atributy je to součet, minimum, maximum, průměr, … a pro kategoriální atributy zase počet různých kategorií, nejčastější kategorie, …).

Pro relace s kardinalitou n:m je to nejsložitější a pokud není k dispozici expert, tak se zvolí jedna tabulka jako hlavní a přistupuje se k ní, jako k relaci 1:n (nebo 1:m). Je možné také přidat agregované informace za druhou část relace, tedy 1:m (nebo 1:n).

## 4.5 Velký počet atributů

Často jsou skutečná data příliš rozsáhlá a v nezměněné podobě by trvalo vytvoření rozhodovacího stromu až příliš mnoho času. Pro rozhodovací strom se doporučuje maximálně kolem 50 atributů. Proto se používají metody ke zmenšení počtu atributů. Bohužel každá z metod má svoje nedostatky. Největším z nedostatků je zhoršení klasifikace rozhodovacího stromu z důvodu právě vyřazení některých atributů (dochází ke ztrátě části informace). Bohužel mohou nastat případy, kdy právě tyto ztracené části informace mají velký vliv na klasifikaci do daných tříd. Základní metody pro snížení počtu atributů jsou:

1) Zahození atributů s malým vlivem na výsledek – viz kapitola 4.5.1
2) PCA (metoda hlavních komponent) – viz kapitola 4.5.2
3) Citlivostní analýza pomocí neuronových sítí – viz kapitola 4.5.3

Podrobněji jsou metody rozebrány v následujících podkapitolách. Tyto metody lze navíc použít k zobrazení dat, kdy se vyberou dva až tři atributy, případně jejich kombinace, která bude výsledkem snižování a ty se použijí k zobrazení dat do 2D nebo 3D grafů. Samozřejmě v případě různých kombinací atributů nebudou grafy jednoduše pochopitelné, ale může v nich být lépe vidět dané rozmístění příkladů do jednotlivých tříd. V případě kombinací atributů, kdy jsou dané atributy určitou mírou korelované, nebude možně jednoduše určit, zda zařazení do dané třídy je z důvodů jednoho nebo více atributů, což v některých případech nemusí být přípustné řešení. Například se bude provádět zjišťování příčiny úmrtí, které budou mít za následek 2 různá onemocnění fungující pouze jako katalyzátory. Příčina se bude hledat ve zdravotních záznamech pacientů (velké množství různorodých vyšetření) a v prostředích, v kterých se pohybovali. Z důvodu velkého počtu atributů se provede například PCA analýza, která požadovaný prostor zmenší a z obou nemocí a stáří pacienta se nakombinuje nový atribut, který rozhodovací strom určí jako hlavní příčinu. Ale který z původních atributů nejvíce přispívá k daným úmrtím? To je potřeba zjistit a při použití PCA analýzy již nejsou tyto informace k dispozici.

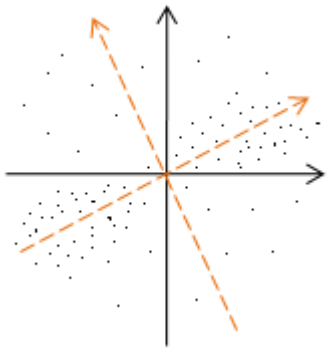
### 4.5.1 Zahození atributů s malým vlivem na výsledek

Nejjednodušší způsob zmenšení počtu atributů je odseknutí málo významných atributů. V této metodě je důležité určit, jak moc velký má daný atribut vliv na klasifikaci výsledku. Může se použít jedna z metod, která se používá při hledání vhodného atributu jako kořene podstromu v algoritmu TDIDT a nebo se použije lineární regrese, která mimo jiné také určí intervaly spolehlivosti.

Atributy s malým vlivem na výsledek, někdy až nežádoucím, jsou například pořadová čísla příkladů v datech a jiné atributy se skoro unikátními hodnotami (například jména, …). Tyto atributy se dají téměř vždy vyřadit bez kontroly jejich vlivu na výsledek.

### 4.5.2 PCA (Principal komponent analysis)

Metoda hlavních komponent, jak se PCA někdy nazývá, slouží k nalezení korelovaných atributů a jejich nahrazení novými nekorelovanými atributy. Fungování metody se dá dobře představit v n-dimenzionálním prostoru (každý atribut představuje jednu osu - dimenzi) jako nahrazení původních os této sousavy novými, které jsou na sebe ortogonální tak, aby body v tomto prostoru (jednotlivé příklady) byly rozprostřeny co nejvíce ve směru nových os (viz obrázek 5). Toto se provádí pouze pomocí operace rotace původních os, která se dá zapsat pomocí matice.



Obrázek 5: Rotace os vycentrovaného 2D systému PCA analýzou.

Nechť vstupní data (příklady) tvoří matice $X$ o velikosti $m \times n$ ($m$ je počet příkladů a $n$ počet atributů). Tato matice se pro správný výsledek musí vycentrovat $Y = X - \bar{X}$ (kde $\bar{X}$ je vektor středních hodnot jednotlivých atributů). Matice závislosti jednotlivých atributů matice $Y$ se spočítá (vzhledem k tomu, že matice $Y$ je už centrovaná) jako:

$$cov(Y) = \frac{1}{m-1} Y^T Y \tag{22}$$

a zároveň platí, že kovarianční matice $cov(A)$ je symetrická (tj. $cov(A) = \big(cov(A)\big)^T$.

Cílem je dostat kovarianční matici, která bude diagonální (tj. závislost dvou různých atributů bude nulová). Mějme výslednou matici Z jako výsledek po rotaci os:

$$Z = Y \times G \tag{23}$$

kde $G$ je matice rotace. Podle toho, že chceme, aby atributy (sloupce) matice $Z$ byly nezávislé, tak platí:

$$konst \times cov(Z) = G^T \times Y^T \times Y \times G = konst_2 \times G^T \times cov(Y) \times G \tag{24}$$

Pro zjištění matice $G$ bude dále uvedeno několik definic tvrzení.

**Definice 5:**

Nechť $A \in C^{n \times n}$ ($C$ je množina komplexních čísel). Jestliže platí $Au = \lambda u$ pro jisté komplexní číslo $\lambda \in C$ a jistý nenulový vektor $u \in C^n, u \neq 0$, potom číslo $\lambda$ se nazývá vlastní číslo matice $A$ a vektor $u$ vlastním vektorem příslušným k tomuto vlastnímu číslu. Množina všech vlastních čísel se nazývá spektrum matice $A$.

**Tvrzení 1:**

Symetrická matice $A \in R^{n \times n}$ má všechna vlastní čísla reálná.

**Tvrzení 2:**

Vlastní vektory příslušné navzájem různým vlastním číslům symetrické matice $A \in R^{n \times n}$ jsou navzájem ortogonální.

Z definice a dvou předešlých tvrzení, které zde nebudou dokázány (viz (Bečvář, 2000)), je vidět, že matice $G$ existuje a odpovídá matici složené s vlastních vektorů kovarianční matice $Y$ (která je symetrická).

Z definice vlastního čísla a kovarianční matice vyplývá, že čím je vlastní číslo větší, tím větší je rozptyl hodnot ve směru, který určuje příslušný vlastní vektor. Proto se v PCA analýze vlastní čísla většinou seřadí podle velikosti a s nimi i odpovídající vlastní vektory. Pokud bude požadováno dále odstranit málo významné atributy (z pohledu dat), tak se to provádí pouhým ořezáním posledních atributů (vlastních čísel a vektorů).

Rozptyl hodnot v daném novém směru může být ale zavádějící, pokud vstupní hodnoty nejsou stejně normovány. Tady může také sehrát důležitou roli šum, který může rozptyly hodně zamíchat. Navíc, i když je variabilita dat v některých směrech malá, tak podíl na klasifikaci může mít velký. Tato problematika je řešena například v práci (Militký, a další), ale i jinde.

Ze způsobu výpočtu závislostí původních atributů je vidět, že se rozpoznává jenom lineární závislost. A to nemusí vždy stačit.

Po provedení klasifikace a zjišťování, které atributy a jaký mají vliv na výsledek je dobré vědět, že zpětný převod hodnot z nové souřadné soustavy do původní se provádí následovně:

$$X' = Z \times G^T + \bar{X} \tag{25}$$

PCA analýza se dá dobře využít k 2D nebo 3D zobrazení původních dat tak, aby bylo dobře vidět například shluky hodnot, vývoj ve směru trendů, atd.

### 4.5.3 Citlivostní analýza pomocí neuronových sítí

Pro citlivostní analýzu a zmenšení vstupního prostoru dat pomocí neuronových sítí se používá metoda komprese, která je popsána v podkapitole 4.5.3.2. Základy o vícevrstvé neuronové síti jsou popsány v podkapitole 4.5.3.1, kde je vysvětleno také učení dané neuronové sítě a několik vlastností, které s tím souvisí.

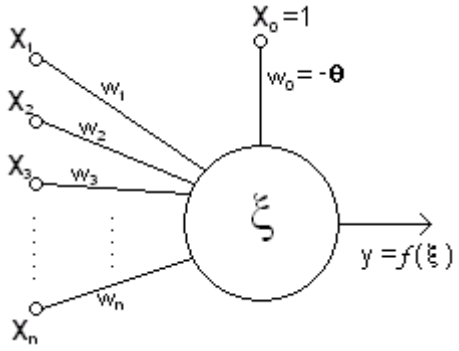
#### *4.5.3.1 Úvod do neuronových sítí*

V této kapitole je popsán obecný formální neuron a vrstevnatá architektura neuronových sítí. Dále se kapitola zabývá speciálním druhem neuronu (perceptronem) a učícím a vybavovacím procesem vícevrstvé perceptronové sítě včetně základních vylepšeních tohoto procesu.

Základem neuronové sítě je umělý neuron. V neuronové síti jsou tyto umělé neurony vzájemně propojeny spoji, které jsou ohodnoceny vahami. Umělý neuron se pomocí změny těchto vah může přizpůsobovat a učit se a tím i celá neuronová síť, v které je zapojen. Jednou z důležitých vlastností neuronových sítí je právě schopnost učení a nacházení závislostí v datech a jejich zevšeobecňování.

Formální neuron (viz obrázek 6) se skládá ze:

1) vstupů $X_{1,..n}$ , ke kterým jsou přiřazeny váhy $w_{1,...n}$. $X_0$ je jenom formální vstup nabývající hodnoty 1. $\theta$ je práh citlivosti neuronu a $w_0$ se nazývá bias ($= -\theta$),
2) $\xi$ je vnitřní potenciál neuronu, který je výsledkem všech vstupů a jejich vah (včetně $w_0$),
3) výstupu, který je výsledkem aktivační funkce na daný vnitřní potenciál $\xi$.




Obrázek 6: Formální neuron

Nejpoužívanějším typem neuronu je perceptron, který pracuje s reálným oborem parametrů. Tento neuron navrhl F.Rosenblatt. Vnitřní potenciál perceptronu se počítá jako vážený součet vstupů:

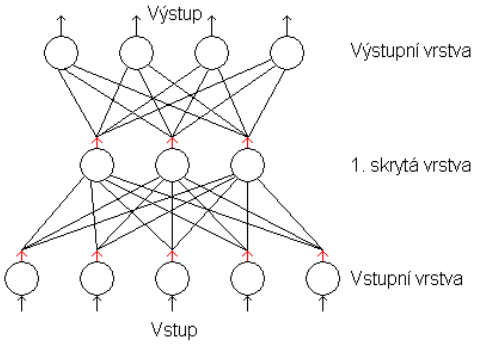
$$\xi = \sum_{i=0}^{n} X_i w_i \tag{26}$$

Aktivační funkce perceptronu je sigmoida:

$$f(\xi) = \frac{1}{1 + e^{-\lambda \xi}} \tag{27}$$

kde $\lambda$ je parametr strmosti

Perceptrony nejčastěji tvoří vícevrstvou neuronovou (perceptronovou) síť (viz obrázek 7).




Obrázek 7: Vícevrstvá neuronová síť

Tato síť je složena z vrstev neuronů. Každý neuron z nějaké vrstvy je vstupy propojen s každým neuronem (jeho výstupem) právě jenom z předchozí vrstvy. Nultá vrstva této sítě je speciálně nadefinovaná jako vstupní a výstupem každého neuronu vstupní vrstvy je hodnota vstupu. Tato vstupní vrstva je jenom formálně nadefinovaná a za normální vrstvu se nepovažuje. Proto vícevrstvá neuronová síť se vstupní vrstvou a X dalšími vrstvami se nazývá X-vrstvá neuronová síť. Poslední vrstva sítě se nazývá výstupní. Vrstvy, které nejsou vstupní ani výstupní se nazývají skryté (každá X-vrstvá neuronová síť má (X-1) skrytých vrstev).

Architektura vícevrstvé neuronové sítě (počet vrstev, počet neuronů v jednotlivých vrstvách) závisí na konkrétní povaze úlohy, kterou má síť řešit. Určení správné architektury je důležité z důvodu schopnosti neuronové sítě se učit a zároveň zobecňovat. Tyto dva požadavky jsou protichůdné. Neuronová síť se učí lépe, čím více má v sobě neuronů a vrstev. Na druhé straně, pokud má síť mnoho vrstev a neuronů, tak dochází k přeučení. Přeučení je stav, kdy se neuronová síť naučila rozpoznávat jenom trénovací data a nová data, která nedostala v procesu učení, rozpoznat nedokáže (nebo spíše dokáže, ale špatně). Schopnost nepřeučit se je vlastně schopnost umět zobecňovat.

Pro stanovené architektury sítě se využívají různé heuristiky a hlavně vlastní zkušenosti s podobnými daty. Jedna z heuristik napovídá, že pro 3-vrstvou perceptronovou síť je dobré použít v první vrstvě o málo více neuronů, než kolik je vstupů. V druhé vrstvě pak tolik neuronů, kolik dá aritmetický průměr počtu neuronů v 1. vrstvě a výstupní vrstě. Pokud je znám charakter úlohy a struktura trénovacích dat, je možné vyjít z vlastností, které umožňuje uspořádání perceptronů. Perceptron má schopnost rozdělit prostor nadrovinou (počítá totiž lineární kombinaci vstupů, což je rovnice nadroviny). Takže první vrstva rozdělí vstupní prostor nadrovinami. Druhá vrstva rozdělí prostor kombinacemi nadrovin (tedy konvexními útvary). Třetí kombinacemi konvexních útvarů (tedy už nekonvexními útvary). Více vrstev se už moc nepoužívá, protože ve většině případů jsou tři maximálně čtyři naprosto postačující a ještě pochopitelné a vizualizovatelné.

Samozřejmě jsou i výjimky, kdy se používá více vrstev. Typický příklad je vícevrstvá neuronová síť, která se používá ke kompresy vstupního prostoru. Taková síť je vlastně síť složená ze dvou neuronových sítí, kompresní a dekompresní. A vzhledem k tomu, že se dopředu nezná, jak má vypadat komprimovaný výstup, tak se síť učí najednou komprimovat i

dekomprimovat a v tom případě už vstup i výstup sítě je dobře znám a tedy se síť může učit identity.

Vícevrstvá perceptronová síť při získávání celkového výsledku funguje následovně:

1) Postupuje se postupně po vrstvách.
2) Nejdříve se spočítají výstupy od všech neuronů z 1. vrstvy. Tím jsou k dispozici všechny vstupy pro 2. vrstvu a může se spočítat výstup od všech neuronů z 2. vrstvy, atd.
3) Tímto způsobem se dopočítá neuronová síť k výsledku (výstupy neuronů z výstupní vrstvy).

Tento postup se nazývá dopředné šíření.

Učení neuronové sítě je snaha nastavení jednotlivých vah neuronů tak, aby byla chyba mezi požadovaným výstupem sítě a dosaženým výstupem minimální. Proto se definuje chyba sítě jako

$$E = \sum_k E_k \ , \tag{28}$$

kde index $k$ představuje pořadové číslo trénovacího vzoru a $E_k$ je chyba odpovídající $k$-tému trénovacímu vzoru a je definovaná vztahem

$$E_k = \frac{1}{2}\sum_j \left(y_{jv} - d_{kj}\right)^2 \ , \tag{29}$$

kde $y_{jv}$ je výstup $j$-tého neuronu výstupní ($v$-té) vrstvy a $d_{kj}$ je požadovaný výstup $j$-tého výstupního neuronu pro $k$-tý trénovací vzor.

V základním modelu sítě se pro optimalizaci používá gradientní metoda, která po získání konečné chyby $E$ upravuje jednotlivé váhy v síti následovně:

$$w_{ijv}(t) = w_{ijv}(t-1) + \Delta w_{ijv}(t) \ , \tag{30}$$

kde $w_{ijv}$ je $i$-tá váha $j$-tého neuronu ve $v$-té vrstvě. Pro přehlednost je $i$-tá váha vázáná na výstup $i$-tého neuronu z $(v-1)$. vrstvy. Dále $t$ značí $t$-tou aktualizaci vah a $\Delta w_{ijv}$ je změna vah získáná jako gradient největšího spádu (pro minimalizaci chyby $E$):

$$\Delta w_{ijv}(t) = -\eta \frac{\partial E(t)}{\partial w_{ijv}(t)}, \tag{31}$$

kde $\eta$ je parameter učení z intervalu $\eta \in \langle 0{,}1 \rangle$.

Váhy neuronů se pro $t = 0$ (tedy na počátku) inicializují na malé náhodné hodnoty se střední hodnotou v okolí 0. Podle heuristiky je doporučována náhodná veličina z intervalu $\langle -2/s\ , 2/s \rangle$, kde $s$ je počet vstupů neuronu, pro který se váha má nastavit.

Pro vícevrstvou perceptronovou síť se učící pravidlo nazývá metoda zpětného šíření (backpropagation). Pomocí gradientní metody se upravují váhy vztahem (30). V tomto vztahu je potřeba zjistit hodnotu $\Delta w_{ijv}$ tedy (podle vztahu

(31)) hodnotu $\frac{\partial E}{\partial w_{ijv}}$. Vzhledem k lineárnosti operátoru derivace a sčítání platí:

$$\frac{\partial E}{\partial w_{ijv}} = \sum_k \frac{\partial E_k}{\partial w_{ijv}}. \tag{32}$$

Podle pravidla o derivaci složené funkce platí:

$$\frac{\partial E_k}{\partial w_{ijv}} = \frac{\partial E_k}{\partial y_{jv}} \frac{\partial y_{jv}}{\partial \xi_{jv}} \frac{\partial \xi_{jv}}{\partial w_{ijv}} \tag{33}$$

$$\frac{\partial \xi_{jv}}{\partial w_{ijv}} = \frac{\partial \sum_i y_{i(v-1)} w_{ijv}}{\partial w_{ijv}} = y_{i(v-1)} \tag{34}$$

$$\frac{\partial y_{jv}}{\partial \xi_{jv}} = \frac{\partial \frac{1}{1+e^{-\lambda \xi_{jv}}}}{\partial \xi_{jv}} = \frac{\lambda e^{-\lambda \xi_{jv}}}{\left(1+e^{-\lambda \xi_{jv}}\right)^2} \tag{35}$$

Ze vlastnosti sigmoidy $\left(1 - \frac{1}{1+e^{-\lambda \xi}}\right) = \frac{e^{-\lambda \xi}}{1+e^{-\lambda \xi}}$ pak vychází:

$$\frac{\partial y_{jv}}{\partial \xi_{jv}} = \lambda \frac{e^{-\lambda \xi_{jv}}}{1+e^{-\lambda \xi_{jv}}} \frac{1}{1+e^{-\lambda \xi_{jv}}} = \lambda \left(1 - y_{jv}\right) y_{jv} \tag{36}$$

$\frac{\partial E_k}{\partial y_{jv}}$ pro výstupní vrstvu:

$$\frac{\partial E_k}{\partial y_{jv}} = y_{jv} - d_{kj} \tag{37}$$

$\frac{\partial E_k}{\partial y_{jv}}$ pro skryté vrstvy:

$$\frac{\partial E_k}{\partial y_{jv}} = \sum_r \frac{\partial E_k}{\partial y_{r(v+1)}} \frac{\partial y_{r(v+1)}}{\partial \xi_{r(v+1)}} \frac{\partial \xi_{r(v+1)}}{\partial y_{jv}} = \sum_r \frac{\partial E_k}{\partial y_{r(v+1)}} \lambda \left(1 - y_{r(v+1)}\right) y_{r(v+1)} w_{r(v+1)} \tag{38}$$

Po dosazení výsledků parciálních derivací (33) až (38) do vztahu (32) zůstává jedině parciální derivace $\frac{\partial E_k}{\partial y_{r(v+1)}}$ nevyřešena. Ale vzhledem k tomu, že tato parciální derivace se musí řešit o vrstvu výše, tak rekurzivně se dá lehce spočítat. Právě pro tuto nutnost počítat a upravovat váhy od výstupní vrstvy postupně směrem ke vstupní dostala metoda svůj název.

Postupů při aktualizaci vah může být více. Základní postup je ten, že se vyhodnotí nejdříve všechna trénovací data a až pak se aktualizují váhy. Při tomto postupu nezáleží na pořadí trénovacích vzorů. Ale nevýhodou je pomalost. K tomuto postupu existuje varianta, že k aktualizaci vah dojde buď hned po každém vyhodnocení trénovacího vzoru, nebo se vytvoří

menší dávky trénovacích dat a ke změně dojde vždy po vyhodnocení chyby přes celou dávku. V těchto variantách už záleží v jakém pořadí (nebo v kterých dávkách) se trénovací vzory dostávají k vyhodnocení neuronovou sítí.

Jednou z nevýhod gradientní metody je to, že při postupném pohybu po chybové funkci se může síť dostat do lokálního minima, které je ale nežádoucí. K této situaci dochází docela často, a proto bylo vyvinuto několik metod k jejímu řešení. Jednou z metod je měnit parametr učení $\eta$, který může hodně ovlivnit chování sítě, hlavně rychlost učení a konvergenci k řešení. Při malém parametru učení se chyba zmenšuje pomalu a při velkém zase může docházet k divergenci sítě. Doporučuje se proto na začátku nastavit parametr učení na velkou hodnotu a postupně tuto hodnotu snižovat.

Nerozšířenějším způsobem předcházení uváznutí v lokálním minimu je přidání do rovnice pro adaptaci vah moment:

$$\Delta w_{ijv}(t) = -\eta \frac{\partial E(t)}{\partial w_{ijv}(t)} + \alpha \Delta w_{ijv}(t-1) \tag{39}$$

kde $\alpha \in (0,1)$ je parametr momentu. Hodnota parametru se volí blízko jedné. Moment představuje setrvačnost, s kterou se síť po chybové funkci pohybuje.

Další metodou předcházení uvíznutí do lokálních minim je přidání šumu do rovnice pro adaptaci vah. Tím umožníme síti dostat se dočasně i na místa s vyšší hodnotou chybové funkce a častokrát tak uniknout z lokálních minim. Velkou výhodou této metody je jednoduchost a malá časová náročnost.

V případě, že jsou vzory, které se neuronová síť učí, velmi podobné, tak budou jejich minima na chybové funkci blízko sebe. Při malém počtu neuronů budou tato minima málo pokrytá a mohou splynout, což zhorší vlastnosti sítě. Přidáním jednoho nebo více skrytých neuronů do sítě lze dosáhnout zjemnění chybové funkce a tím lépe tato minima rozpoznat. Současně s přidáním nového neuronu se doporučuje snížit hodnotu parametru učení.

#### *4.5.3.2 Citlivostní analýza*

Ke zkoumání závislostí jednotlivých atributů ve vstupních datech a ke zmenšení jejich počtu se dá také využít neuronových sítí. Provádí se to pomocí kompresních neuronových sítí, jejichž architektura je taková, že vstupní a výstupní vrstva má tolik neuronů, kolik je atributů a ve skrytých vrstvách existuje úzké hrdlo (tj. vrstva, která má požadovaný menší počet neuronů než vstupní a výstupní vrstva). Po naučení této neuronové sítě identit (tj. co je zadáno na vstupu, je požadováno na výstupu) je možné tuto síť rozdělit v nejužší vrstvě a tím rozdělit síť na kompresní a dekompresní síť. Výstupy z kompresní sítě se použijí jako nové hodnoty nových atributů. Pro zpětnou transformaci nových atributů do původních se pak použije druhá dekompresní síť.

Standardně se architektura sítě volí jako dvouvrstvá (1 skrytá vrstva) nebo čtyřvrstvá (3 skryté vrstvy), u které 1. skrytá vrstva slouží jako kombinační a 3. jako dekombinační.

Výhodou vícevrstvých perceptronových neuronových sítí oproti PCA je, že nalézají i nelineární závislosti mezi atributy. Nevýhodou ale zase to, že se ještě více stíží schopnost pochopení, proč vlastně byl výsledný rozhodovací strom postaven tak, jak byl postaven. A tím se samozřejmě zhoršuje schopnost extrakce znalostí.

Existují i jiné neuronové sítě, které umožňují kompresy. Například lineární asociativní neuronovou sítí a hebbovským učením lze realizovat PCA. Viz například (Kung, 1993).

### 4.6 Chybějící hodnoty atributů

V reálných datech nejsou často uloženy hodnoty pro všechny atributy. Může se to stát z různých důvodů jako například nedostatek času, šetření s penězi (drahé vyšetření) nebo také z důvodu možné kompromitace a ještě velkého množství jiných důvodů. Pro klasifikaci může být i informace o chybějící hodnotě důležitá. S těmito hodnotami se dá vypořádat při vytváření rozhodovacího stromu různě:

1) Úplně vynechat příklady, v kterých jsou chybějící hodnoty (ale v praxi je takových případů hodně a nelze podstatnou část příkladů ignorovat).
2) Nahrazení chybějící hodnoty:
    a) Expertem.
    b) Nejpravděpodobnější hodnotou (střední hodnota nebo jiné statistické hodnoty. Nejlepší výsledky pro numerické atributy jsou dosahovány s hodnotou, která nemění směrodatnou odchylku).
    c) Úplně novou hodnotou vyhrazenou pro neznámou hodnotu.
    d) Náhodnou hodnotou (může být vygenerována na základě pravděpodobností).

Pokud se jedná o případ, kdy se má daný příklad klasifikovat, tak se může použít jedna z metod nahrazení chybějící hodnoty nebo se může provést vícenásobný průchod rozhodovacím stromem pro každou přípustnou možnost nebo jenom pár nejpravděpodobnějších možností a výsledek může být buď tabulka s jakou pravděpodobností spadá příklad do dané třídy, nebo se vybere jenom ta nejpravděpodobnější třída. Vždy záleží na tom, co je požadováno.

# 5 Extrakce znalostí, jejich reprezentace, vizualizace a interpretace

Důležitou součástí dobývání znalostí je i extrakce nalezených znalostí a také jejich vizuální zobrazení. Tuto problematiku rozebírají následující kapitoly.

## 5.1 Extrakce znalostí z rozhodovacích stromů

Tyto znalosti jsou různé a závisí na typu dat. Před každým vytvořením rozhodovacího stromu a jeho otestováním se musí určit, s jakou spolehlivostí se očekává klasifikace popřípadě jaká je přípustná chyba (viz kapitola 6). Pokud rozhodovací strom nesplní tyto základní požadavky, tak může nastat jedna z těchto variant (nebo jejich kombinace):

1) data nebyla správně předzpracována,
2) úloha není určena pro zpracování rozhodovacím stromem,
3) chybí další informace, které jsou relevantní ke klasifikaci,
4) nároky na schopnost klasifikace jsou příliš vysoké, nebo
5) data jsou příliš zašuměná.

Zjištění, které varianty jsou příčinou špatných výsledků, je obtížné a je k tomu potřeba hlavně čas. Některé varianty se dají zjistit relativně snadno. Například zkusit jinou metodu dobývání znalostí jako jsou neuronové sítě, ILP (induktivní logické programování) a jiné. Pokud ani jedna z metod dobývaní znalostí nedává výrazně lepší výsledky, tak je potřeba se znovu zamyslet nad správným předzpracováním dat. Pokud ani to nepomáhá, pak po zvážení nároků na schopnost klasifikace je potřeba doufat a hledat další informace, které by byly relevantní pro klasifikaci, protože z příliš zašuměnými daty se toho už moc dělat nedá.

V případě, že klasifikační schopnost rozhodovacího stromu vyhovuje požadavkům, je možné přejít k extrakci dalších znalostí (to, že data je možné klasifikovat rozhodovacím stromem může být samo o sobě podstatná znalost). Nejjednodušší znalost, kterou rozhodovací strom může dát, je informace o tom, které atributy nemají nebo mají minimální vliv na klasifikaci. Toto může být důležitá znalost například při úspoře finančních prostředků, pokud atributy představují výsledky nějakých testů, které něco stojí.

Další znalosti o vlivu atributů na klasifikaci je možné lépe poznat ze správné vizualizace nalezených znalostí viz následující kapitola. Zajímavý způsob je převod rozhodovacího stromu na pravidla, která se dají setřídit a sama pak ještě klasifikovat podle důležitosti nebo jiným způsobem.

## 5.2 Reprezentace a vizualizace znalostí

Nedílnou součástí vytváření rozhodovacích stromů a jejich dalšího použití je vizualizace nalezených znalostí. Bohužel neexistuje jeden způsob, jak tyto znalosti srozumitelně zobrazit. Pro každého je srozumitelnější něco jiného a pro různé druhy dat jsou vhodnější jiné způsoby zobrazení. Proto v následujících podkapitolách bude ukázáno několik způsobů vizualizace znalostí.

### 5.2.1 Formátovaný text

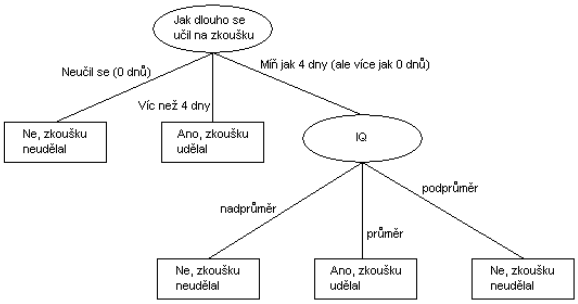
Strom se vypisuje postupně procházením do hloubky. Každá úroveň je odsazená víc než ta předešlá. Každý uzel obsahuje informace o své identifikaci, hodnotě atributu, která představuje větev od rodiče, a další volitelné hodnoty (například poměry zastoupení příkladů v jednotlivých třídách). Pro listy se přidává na konec informace o tom, že je daný uzel list (například „*“). Je to jednoduchý způsob, který nepotřebuje grafické prostředí. Je ale málo čitelný pro větší stromy, viz následující příklad:

Uzel 1: Zastoupení tříd: N = „Ano“:„Ne“ = 5:3
    Uzel 2: „Jak dlouho se učil“ = „Neučil se“ ; N = 0:1 *
    Uzel 3: „Jak dlouho se učil“ = „Víc než 4 dny“ ; N = 3:0 *
    Uzel 4: „Jak dlouho se učil“ = „Míň jak 4 dny“ ; N = 2:2
        Uzel 5: „IQ“ = „nadprůměr“ ; N = 0:1 *
        Uzel 6: „IQ“ = „průměr“ ; N = 2:0 *
        Uzel 7: „IQ“ = „podprůměr“ ; N = 0:1 *

Lépe čitelná je varianta, kdy je strom v textové formě zobrazován v nějakém vizuálním nástroji, který obarvuje různé úrovně, má možnost schovávat podstromy (formou adresářů nebo jenom „+“ na začátku každého uzlu) a jiné užitečné schopnosti.

### 5.2.2 Strom

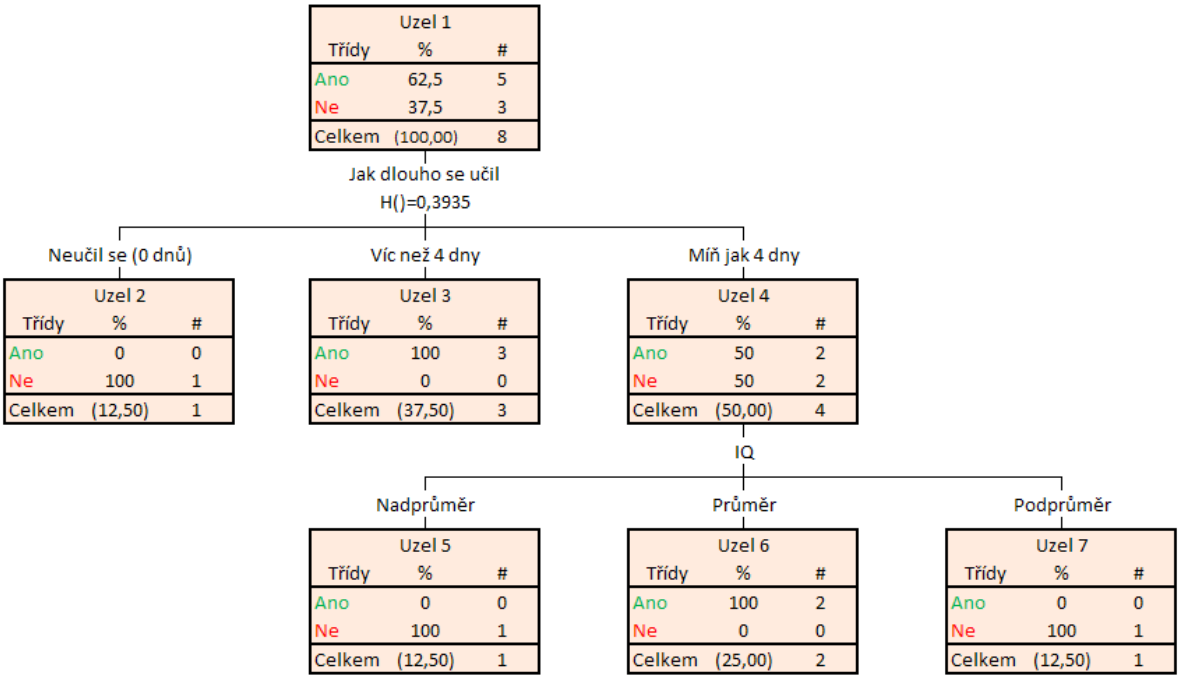
Rozhodovací strom, jak už název napovídá, je stromová struktura a jako taková se dá lehce vizualizovat v podobě stromu (viz obrázek 8). Tento typ zobrazení je dobře srozumitelný pokud jde o ne příliš košatý strom (jakmile se strom nedá zobrazit jako celek s dobře čitelnými popisky).




Obrázek 8: Příklad rozhodovacího stromu zobrazeného formou stromu.

### 5.2.3 Dendrogram

Dendrogram (viz obrázek 9) se podobá stromu, ale zobrazuje kromě samotné stromové struktury také informace o rozdělení příkladů v daných uzlech a listech. Může obsahovat různé hodnoty, které byly spočteny pro výběr daného uzlu jako dalšího reprezentanta nového podstromu a jiné.




Obrázek 9: Dendrogram

### 5.2.4 Obecný logický diagram

Obecný logický diagram je tabulka, která obsahuje všechny možné příklady dat, tedy všechny kombinace všech hodnot atributů. Výsledek klasifikace rozhodovacím stromem je potom znázorněn v tabulce barvou nebo jiným podobným způsobem a správnost klasifikace s informací od učitele kolečkem nebo křížkem (viz obrázek 10). Tento způsob navrhl Michalski (Michalski, 1978). Bohužel tato metoda je vhodná a srozumitelná jenom pro malý počet atributů.

| IQ | Je oblíbený | | | |
|---|---|---|---|---|
| Průměrné | Ano | O | O | O |
| | Ne | O | O | X |
| Nadprůměrné | Ano | X | X | O |
| | Ne | O | O | O |
| Podprůměrné | Ano | X | O | O |
| | Ne | O | O | O |
| **Jak dlouhu se učil na zkoušku** | | Víc než 4 dny | Míň jak 4 dny | Neučil se |

Obrázek 10: Obecný logický diagram

### 5.2.5 Koláčový graf

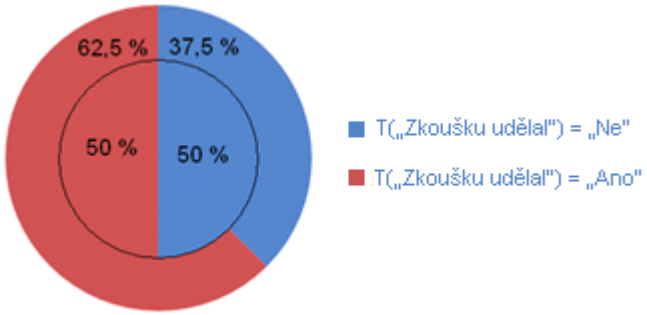
Koláčový graf se používá k zachycení, jak moc dobře dokáže pravidlo charakterizovat klasifikaci příkladů do tříd. Používá se pro lepší názornost. Jako příklad je dále uveden koláčový graf pro pravidlo z příkladu 1 po oříznutí:

IF H(„Jak dlouho se učil na zkoušku“) = „Méně než 4 dny“ THEN T(„Zkoušku udělal“) = „Ne“

Kontingenční tabulka k tomuto pravidlu je:

| | T(„Zkoušku udělal“) = „Ne“ | T(„Zkoušku udělal“) <> „Ne“ | |
|---|---|---|---|
| H(„Jak dlouho se učil na zkoušku“) = „Méně jak 4 dny“ | 2 | 2 | 4 |
| H(„Jak dlouho se učil na zkoušku“)<>„Méně jak 4 dny“ | 1 | 3 | 4 |
| | 3 | 5 | 8 |

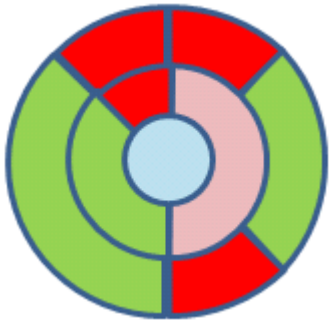
Z čehož pak vyplývá, že dané pravidlo má platnost 2/4. Relativní četnost třídy T(„Zkoušku udělal“) = „Ne“ je 3/8. Z toho pak vyplývá, že pravidlo dobře charakterizuje, kdy student zkoušku neudělá. Obrázek 11 zobrazuje tento koláčový graf.




Obrázek 11: Koláčový graf jednoho pravidla

### 5.2.6 Výsečový graf

Výsečový graf se podobá koláčovému, ale zobrazuje rozvětvení rozhodovacího stromu od kořene až do listů. Každý prstenec reprezentuje jednu hladinu stromu. Kořen je reprezentován vnitřním kruhem. Vnější prstenec je rozdělen do tolika částí, kolik je celkem ve stromu listů. Každá výseč prstence odpovídá velikostně procentnímu podílu příkladů (podobně jako u koláčového grafu).



Obrázek 12: Výsečový graf stromu z obrázku 8

### 5.2.7 Pravidla

Další možností je převést rozhodovací strom na pravidla nebo podmínky. Pravidla mají sice horší vizuální podobu, ale mají výhodu v tom, že se s nimi dá pracovat samostatně.

Převod rozhodovacího stromu na pravidla se provádí následujícím způsobem:

1. Pro každou cestu od kořene do všech listů rozhodovacího stromu vytvoř pravidlo podle bodu 1a a 1b.
   1a. Pro každou hranu, kterou prochází daná cesta vytvoř jeden člen podmínky, která odpovídá dané hraně (tj. H(atribut) = hodnota) a tento člen svaž s ostatními konjukcí.
   1b. Výsledek pravidla nastav na odpovídající klasifikaci, kterou reprezentuje daný list.

Například rozhodovací strom z obrázku 8 po převodu na pravidla vypadá následovně:

1. IF (H(„Jak dlouho se učil na zkoušku“)==“Neučil se“) THEN T(„Zkoušku udělal“) = „Ne“
2. IF (H(„Jak dlouho se učil na zkoušku“)==“Víc než 4 dny“) THEN T(„Zkoušku udělal“) = „Ano“
3. IF (H(„Jak dlouho se učil na zkoušku“)==“Míň než 4 dny“)
   && (H(IQ)==”Průměrné“) THEN T(„Zkoušku udělal“) = „Ano“
4. IF (H(„Jak dlouho se učil na zkoušku“)==“Míň než 4 dny“)
   && (H(IQ)==”Nadprůměrné“) THEN T(„Zkoušku udělal“) = „Ne“
5. IF (H(„Jak dlouho se učil na zkoušku“)==“Míň než 4 dny“)
   && (H(IQ)==”Podprůměrné“) THEN T(„Zkoušku udělal“) = „Ne“

Tyto pravidla se dále dají seřadit podle schopnosti správně klasifikovat nebo jiné důležitosti (popřípadě i schopnosti minimalizovat cenu, za špatnou klasifikaci).

Navíc se dají zběžně rozdělit na ty, které jsou mezi experty dobře známé, dále ty, které nejsou tak jasné a známé a experti se je s velkou pravděpodobností naučí, a nakonec ty, které jsou tak málo významné nebo tak komplikované a závislé na dalších faktorech, že se shrnou do bodu, kdy se experti budou muset poradit s počítačem. Typicky vhodné použití tohoto rozdělení by mohlo být například v medicíně, kdy se lékař nemůže každou chvíli bavit s počítačem, ale musí se rozhodovat okamžitě. Rozhodovacím stromem by se dalo toto chování reprezentovat novou třídou „Poraď se dál“, kterou bychom mohli zmenšit a zpřehlednit původní rozhodovací strom.

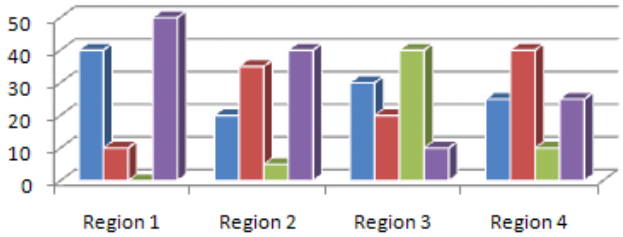
Samozřejmě každé rozdělení pravidel je vázáno na dané potřeby a pro různá data (obory) se provádí specifické rozdělení pro experty.

### 5.2.8 Další možnosti zobrazení

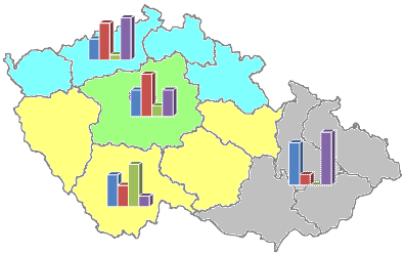
Dalšími možnostmi zobrazení dat a jejich klasifikace pomocí rozhodovacích stromů může být velmi variabilní a také může, ale také nemusí, záležet na některých atributech, které byly k dispozici při vytváření rozhodovacího stromu.

Jeden z příkladů může být zobrazení předpokládaných ztrát či zisků za jednotlivé regiony. Samozřejmě se dají výsledky zobrazit sloupcovými a jinými grafy namačkanými vedle sebe, ale když se propojí i s mapou, kde budou výsledky zobrazeny v odpovídajících částech, může tím zobrazení dávat další informace, které mohou být zajímavější (viz na porovnání obrázek

13 a obrázek 14), ale může to být zase na úkor přesnosti (například bez informací o orientačních počtech).




Obrázek 13: Sloupcové grafy



Obrázek 14: Sloupcové grafy v mapě

# 6 Testování a vyhodnocování výsledků

Důležitou součástí dobývání znalostí je také testování nalezených znalostí. Způsob testování je dobré si rozmyslet ještě před samotným dobýváním (vytvořením rozhodovacího stromu). Důležitá věc je, na jakých datech se bude testování provádět a na co se bude klást při testování důraz. Tyto věci jsou řešeny v následujících podkapitolách.

## 6.1 Rozdělení dat na testovací a trénovací

Ještě před samotným vytvářením rozhodovacího stromu je potřeba mít připravena data, na kterých se bude schopnost klasifikace stromu testovat. V případě, že se vytváří více možných rozhodovacích stromů, je možné mít ještě navíc data validační. V tom případě testovací data budou sloužit jako parametr výběru jednoho z vytvořených rozhodovacích stromů a validační pak vyhodnotí daný výběr. Validační data se často vynechávají, pokud se nejedná o porovnávání dvou zcela různých přístupů nebo modelů pro dobývání znalostí.

Testování rozhodovacích stromů a i obecně modelů, které využívají učení s učitelem (tj. jsou k dispozici trénovací data u kterých je znám požadovaný výsledek), je možné rozdělit do několika variant, podle toho, jaká data se použijí pro učení a jaká pro testování. Tyto varianty jsou:

1) testování na testovacích datech
2) testování v celých trénovacích datech
3) bootstrap
4) křížová validace
5) leave-one-out

### 6.1.1 Testování na testovacích datech

Testování na testovacích datech je nejjednodušší, ale ne vždy je možné. Toto testování předpokládá, že existují trénovací data a testovací data. Ve skutečnosti toto rozdělení dat je málokdy k dispozici. Častěji nastává případ, že z dat je vybrána množina pro vytvoření rozhodovacího stromu a testování se provádí jednou z dalších variant.

### 6.1.2 Testování v celých trénovacích datech

Testování v celých trénovacích datech má velmi malou vypovídající schopnost, protože vůbec nic neříká o schopnosti rozhodovacího stromu zobecňovat naučenou klasifikaci. Může se často stát, že strom bude přeučený. Proto se z dat pro trénování a vytvoření rozhodovacího stromu vyberou dvě skupiny. Jedna skupina bude sloužit pro trénování a druhá pro testování. V tomto případě je důležité, jak se obě skupiny vytvoří (viz dále).

### 6.1.3 Bootstrap

Bootstrap vytváří testovací skupinu dat náhodným výběrem z dostupných dat a do testovací skupiny zařadí všechny nevybrané příklady do trénovací skupiny. Pokud je k dispozici $n$ příkladů, tak se provede $n$-krát náhodný výběr ze všech těchto příkladů a vybrané se přidají

do trénovacích dat, tj. trénovací data budou obsahovat duplicity. Pravděpodobnost, že nějaký příklad vybrán nebude je $1 - \left(\frac{1}{n}\right)$. Limitní pravděpodobnost, že příklad vybrán nebude po $n$ náhodných výběrech je:

$$\lim_{n\to\infty}\left(1-\left(\frac{1}{n}\right)\right)^{2} = e^{-1} \cong 0{,}367879441\,.$$

Takže pro rozumně velká data se vybere přibližně 63,21% příkladů k trénování a 36,79% k testování. Empiricky doporučený poměr trénovacích dat k testovacím je 75:25. U bootstrap se může stát, že v případě malého zastoupení příkladů zařazovaných do nějaké třídy, nemusí být vybráno do trénovacích dat dostatečné zastoupení dané třídy. Proto se občas provádí výběr trénovacích dat tak, že se dostupná data rozdělí do skupin podle odpovídajících tříd a až z těchto tříd se provádí náhodný výběr (tolik náhodných výběrů z dané skupiny, kolik obsahuje příkladů). Tento postup zaručuje stejné zastoupení tříd v trénovací skupině, jako byl v dostupných datech. Limitně pak i stejný poměr v testovacích datech. Někdy je ale nežádoucí tento poměr zachovat, protože může ovlivnit výsledek (budou převažovat příklady z jedné nebo pár vybraných tříd). Proto se občas provádí výběr tak, že se z každé třídy vybere stejný počet náhodných příkladů.

### 6.1.4 Křížová validace

Křížová validace rozdělí dostupná data nejdříve na několik $n$ stejně velkých částí. Pak se vždy jedna část vyjme a použije se pro testování a zbylé se použijí pro trénování. Toto se provede celkem $n$-krát, aby se pro testování použily postupně všechny části. Výsledek testu (viz kapitola 6.2 Vyhodnocení testů) se pak zprůměruje. Tak jako pro bootstrap se i zde dá využít rovnoměrnější zastoupení jednotlivých tříd v jednotlivých částech, aby byla zaručena lepší spolehlivost. Křížové validaci, která rozděluje data do $n$ částí, se říká $n$-násobná křížová validace.

### 6.1.5 Leave-one-out

Leave-one-out je variantou křížové validace, kdy počet částí, do kterých se data rozdělí, je roven počtu všech příkladů. Takže pro testování se vybere vždy jenom jeden příklad. Tato varianta testování dává odhad, jak by se znalosti získané ze všech $n$ testováních chovaly při klasifikaci neznámých příkladů. Nevýhodou této varianty je, že na první pohled pro tisíce příkladů, které jsou k dispozici, by muselo proběhnout tisíce vytváření nových stromů a jejich otestování jedním příkladem. Samozřejmě se dá Leave-one-out naimplementovat lépe a už při vytváření stromu s touto variantou testování počítat. Klade to sice vyšší nároky na paměť, ale čas potřebný na testování se tím zkrátí.

Všechny vyjmenované varianty mají něco společného. Neříkají nic o tom, co je výsledkem testu, ale jenom o tom, které příklady vybrat pro testování. V následující kapitole jsou rozepsány způsoby vyhodnocení testů a jejich rozdělení.

## 6.2 Vyhodnocení testů

Cílem testování je určit, kolikrát rozhodovací strom (nebo obecně model) klasifikoval příklady správně a kolikrát se s učitelem při klasifikaci neshodnul. Tyto informace jsou často zachycovány do matice záměn (viz Tabulka 3).

V této matici jsou zaznamenány počty správných a nesprávných zařazení příkladů do daných tříd. $P_{i,i}$ je počet správně klasifikovaných příkladů do i-té třídy. $P_{i,j}$ kde $i \neq j$ je počet špatně klasifikovaných příkladů z j-té třídy podle učitele do i-té rozhodovacím stromem.

| | Klasifikace rozhodovacím stromem | | |
|---|---|---|---|
| Klasifikace učitelem | Třída 1 | Třída 2 | Třída 3 |
| Třída 1 | $P_{1,1}$ | $P_{2,1}$ | $P_{3,1}$ |
| Třída 2 | $P_{1,2}$ | $P_{2,2}$ | $P_{3,2}$ |
| Třída 3 | $P_{1,3}$ | $P_{2,3}$ | $P_{3,3}$ |

Tabulka 3: Matice záměn

V matici záměn jsou uvedeny jenom počty správně, či nesprávně zařazených příkladů, ale v mnoha případech je důležitý i typ chyby. Například pokud se rozhodovací strom dopustí chyby v určení diagnózy pacienta a místo nutné operace ho pošle domů, je to mnohem závažnější chyba, než když pacienta pošle na operaci, i když to nepotřebuje (samozřejmě také to je ale chyba). Různé závažnosti chyb je možné do testování, ale i učení zahrnout formou matice cen. V ní jsou uvedeny ceny za jednotlivé chyby (čím větší chyba tím vyšší cena).

Z matice záměn se dále dá spočítat správnost klasifikace, přesnost a další charakteristiky, které se už dají lehce porovnávat s výstupy jiných rozhodovacích stromů nebo úplně jiných modelů. Tyto charakteristiky jsou popsány v následujících podkapitolách.

### 6.2.1 Správnost klasifikace

Nejjednodušší charakteristiky toho, jak jsou získané znalosti kvalitní, jsou celková správnost nazývaná také úspěšnost nebo celková chyba. Celková správnost je pravděpodobnost, že klasifikované příklady se klasifikovali správně:

$$Acc = \frac{\sum_{i=1}^{n_t} P_{i,i}}{\sum_{i=1}^{n_t} \sum_{j=1}^{n_t} P_{i,j}}, \tag{40}$$

kde $n_t$ je celkový počet klasifikačních tříd.

Celková chyba je procento špatně klasifikovaných příkladů ze všech příkladů v testu. Někdy se k špatné klasifikaci přidává i váha závažnosti dané chybné klasifikace:

$$Err = \frac{\sum_{j=1}^{n_t} \sum_{i=1|i\neq j}^{n_t} P_{i,j} c_{i,j}}{\sum_{i=1}^{n_t} \sum_{j=1}^{n_t} P_{i,j}}, \tag{41}$$

kde $c_{i,j}$ je cena za špatnou klasifikaci příkladů z j-té třídy do i-té třídy normovaná na interval $\langle 0,1 \rangle$.

Někdy je ale vhodnější sledovat správnost či chybu za jednotlivé třídy, protože celková správnost může být zkreslená (hlavně pokud je velmi nerovnoměrné rozložení příkladů do daných tříd). V tom případě se správnost klasifikace dané třídy spočítá:

$$Acc(i) = \frac{P_{i,i}}{\sum_{j=1}^{n_t} P_{i,j}} \tag{42}$$

a chyba klasifikace dané třídy:

$$Err(i) = \frac{\sum_{j=1|j\neq i}^{n_t} P_{i,j} c_{i,j}}{\sum_{j=1}^{n_t} P_{i,j}}. \tag{43}$$

### 6.2.2 Úplnost a přesnost klasifikace

Tyto charakteristiky jsou převzaty z oblasti vyhledávání informací. Úplnost je pravděpodobnost, že příklady z dané třídy byly klasifikované rozhodovacím stromem do správné třídy. Přesnost klasifikace je pravděpodobnost, že příklady klasifikované do dané třídy opravdu z této třídy jsou (totéž co správnost).

$$\acute{U}plnost(j) = \frac{P_{j,j}}{\sum_{i=1}^{n_t} P_{i,j}} \tag{44}$$

$$P\check{r}esnost(i) = \frac{P_{i,i}}{\sum_{j=1}^{n_t} P_{i,j}} \ \left(= Acc(i)\right) \tag{45}$$

Někdy se používá souhrnná statistika *F-míra* úplnosti a přesnosti klasifikace třídy $i$:

$$F(i) = \frac{P\check{r}esnost(i)\ \acute{U}plnost(i)}{\frac{P\check{r}esnost(i) + \acute{U}plnost(i)}{2}} = \frac{P_{i,i}{}^2}{\frac{P_{i,i}\left(\sum_{j=1}^{n_t} P_{i,j} + \sum_{j=1}^{n_t} P_{j,i}\right)}{2}} = \frac{2P_{i,i}}{\sum_{j=1}^{n_t} P_{i,j} + \sum_{j=1}^{n_t} P_{j,i}} \tag{46}$$

### 6.2.3 Specificita a senzitivita klasifikace

Tyto charakteristiky jsou přebrané z medicíny. Senzitivita je pravděpodobnost, že příklady z dané třídy jsou klasifikované modelem (rozhodovacím stromem) do správné třídy (totéž co úplnost). V medicíně slouží například k informaci, jak moc dobře nějaký lék zabírá na nemocné pacienty. Specificita je pravděpodobnost, že příklady z jiných tříd (podle učitele) se neklasifikují špatně do vybrané třídy. V medicíně se používá například k informaci, zda lék zabírá pouze na danou chorobu.

$$Senzitivita(j) = \frac{P_{j,j}}{\sum_{i=1}^{n_t} P_{i,j}} \left(= Úplnost(j)\right) \tag{47}$$

$$Specificita(k) = \frac{\sum_{i=1|i\neq k}^{n_t} \sum_{j=1|j\neq k}^{n_t} P_{i,j}}{\sum_{i=1}^{n_t} \sum_{j=1|j\neq k}^{n_t} P_{i,j}} \tag{48}$$

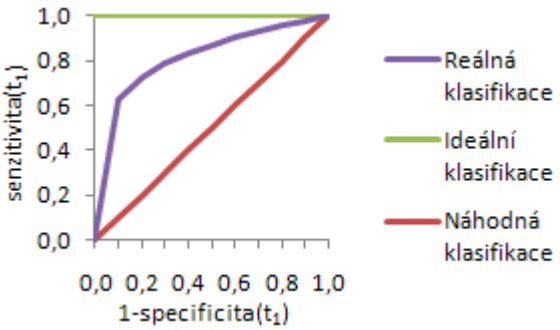
Senzitivita a Specificita se můžou uvádět také dohromady jako vzájemný součin nebo součet. Například Yondenův index je definován jako:

$$YI(i) = senzitivita(i) + specificita(i) - 1 \tag{49}$$

### 6.2.4 Křivka ROC

Křivka ROC slouží k lepšímu porovnání dvou klasifikačních modelů, které zařazují příklady do dvou tříd. Dává do souvislosti senzitivitu (osa y) a (1 - specificitu) (osa x). Křivka se vytváří pomocí různého nastavení prahu, který určuje, zda příklad bude zařazený do jedné nebo druhé třídy. V předešlém textu se uvažoval práh 0,5, tj. pokud v listu rozhodovacího stromu nezůstaly trénovací příklady jenom z jedné třídy, tak byla vybrána první klasifikační třída tehdy, pokud četnost příkladů zařazených do této třídy byla větší, než četnost příkladů zařazených do druhé třídy (četnost příkladů první třídy byla větší nebo rovna 50%). ROC křivka ukazuje, jak se klasifikační model chová při změně tohoto prahu.

Ideální stav je, kdy senzitivita i specificita je 100% (tj. rovna 1). Takže ideální bod v grafu je [0,1], ke kterému se ROC křivka snaží přiblížit. K porovnání ROC křivek je také dobré vědět, že náhodná klasifikace pomocí daných prahů odpovídá úsečce z bodu [0,0] do bodu [1,1] (pro daný práh $\theta$ je: $senzitivita(t_1) = p(t_1) = \theta$ a $\left(1 - specificita(t_1)\right) = \left(1 - (1-\theta)\right) = \theta$). Pokud křivka ROC klasifikačního modelu klesne pod ROC křivku náhodné klasifikace, tak klasifikační model není úplně v pořádku. Pokud je křivka ROC celá pod křivkou náhodné klasifikace, tak nejspíše došlo jenom k prohození klasifikace do daných tříd a stačí je v klasifikačním modelu (v listech rozhodovacího stromu) zaměnit.




Obrázek 15: Křivka ROC pro Ideální, náhodnou a nějakou reálnou klasifikaci

Křivka ROC je primárně určena k zobrazení schopnosti modelu klasifikovat příklady do 2 tříd. Pro modely, které klasifikují příklady do více tříd, se dá vytvořit více ROC křivek (pro každou třídu jedna). Toto může být zajímavé v případě, kdy jeden model klasifikuje některé třídy lépe a některé hůře. Rozdíly mezi těmito odpovídajícími ROC křivkami mohou dát lepší

pohled na schopnosti modelů. Na porovnání dvou klasifikačních modelů pomocí ROC křivek se používají různé metriky. Nejpoužívanější jsou:

1) vzdálenost křivky od bodu [0,1],
2) obsah pod křivkou ($AUC$ = Area Under Curve),

Křivku ROC lze využít i při postupném ořezávání rozhodovacího stromu nebo odebírání atributů, které nemají velký vliv na danou klasifikaci. Při tomto postupu je snaha tuto křivku ROC příliš nezhoršit.

Výhodou křivek ROC je, že zachycují chování modelů bez ohledu na rozdělení tříd a cen (v případě klasifikace jenom do dvou tříd). Například očekávána cena klasifikace modelu v bodě [x, y] křivky ROC (pro třídu $k$) je:

$$Cena = p_{test}(k)\sum_{i=1}^{n_t} c_{i,k}p_{chyba}(i,k) + \big(1 - p_{test}(k)\big)\sum_{j=1}^{n_t} c_{k,j}pp_{chyba}(k,j) \tag{50}$$

kde $c_{i,j}$ je cena za chybnou klasifikaci příkladu z třídy $j$ do třídy $i$ $\big(\forall l: c_{l,l} = 0\big)$,

$p_{test}(k)$ je pravděpodobnost příkladů z třídy $k$ v testovacích datech,

$p_{chyba}(i,k)$ je pravděpodobnost, že příklad z třídy $k$ byl klasifikován do $i$-té třídy.

$$p_{chyba}(i,k) = \frac{P_{i,k}}{\sum_{l=1}^{n_t} P_{l,k}},$$

$pp_{chyba}(k,j)$ je pravděpodobnost, že příklad klasifikovaný modelem do $k$-té třídy patří do třídy $j$. Pravděpodobnost je v rámci příkladů, které nepatří do třídy $k$ (podle učitele).

$$pp_{chyba}(k,j) = \frac{P_{k,j}}{\sum_{l=1|l\neq k}^{n_t}\sum_{m=1}^{n_t} P_{m,l}},$$

Chyba klasifikace příkladů z jiných tříd než $k$ do tříd ostatních kromě $k$ jsou pro danou třídu nezajímavé.

Dva body [x,y], [x‘,y‘] na křivkách ROC budou odpovídat stejně dobrým modelům, pokud budou jejich ceny stejné. Pokud byly oba modely vytrénované na stejné trénovací množině a ceny za jednotlivé chyby jsou stejné, pak:

$$p_{test}(k)\sum_{j=1}^{n_t} c_{j,k}\left(\frac{P_{j,k}}{\sum_{l=1}^{n_t} P_{l,k}} - \frac{P'_{j,k}}{\sum_{l=1}^{n_t} P'_{l,k}}\right) = \big(1 - p_{test}(k)\big)\sum_{i=1}^{n_t} c_{k,i}\left(\frac{P'_{k,i}}{\sum_{l=1|l\neq k}^{n_t}\sum_{m=1}^{n_t} P'_{m,l}} - \frac{P_{k,i}}{\sum_{l=1|l\neq k}^{n_t}\sum_{m=1}^{n_t} P_{m,l}}\right)$$

Pokud navíc platí, že cena za chybnou klasifikaci do třídy $k$, je pro všechny třídy stejná a cena za chybnou klasifikaci příkladů z třídy $k$ do ostatních tříd je také stejná (platí vždy při klasifikaci jenom do dvou tříd), pak:

$$\frac{\big(1 - p_{klasifikace}(k)\big)\,c_{k,l\neq k}}{p_{klasifikace}(k)c_{l\neq k,k}} = \frac{\dfrac{\sum_{j=1|j\neq k}^{n_t} P_{j,k}}{\sum_{j=1}^{n_t} P_{j,k}} - \dfrac{\sum_{j=1|j\neq k}^{n_t} P'_{j,k}}{\sum_{j=1}^{n_t} P'_{j,k}}}{\dfrac{\sum_{i=1|i\neq k}^{n_t}\big(P'_{k,i}\big)}{\sum_{l=1|l\neq k}^{n_t}\sum_{m=1}^{n_t} P'_{m,l}} - \dfrac{\sum_{i=1|i\neq k}^{n_t}\big(P_{k,i}\big)}{\sum_{l=1|l\neq k}^{n_t}\sum_{m=1}^{n_t} P_{m,l}}}. \tag{51}$$

Vzhledem k tomu, že platí:

$$x = 1 - Specificita(k) = \frac{\sum_{i=1|i\neq k}^{n_t} P_{k,i}}{\sum_{i=1}^{n_t}\sum_{j=1|j\neq k}^{n_t} P_{i,j}} \quad \text{a} \tag{52}$$

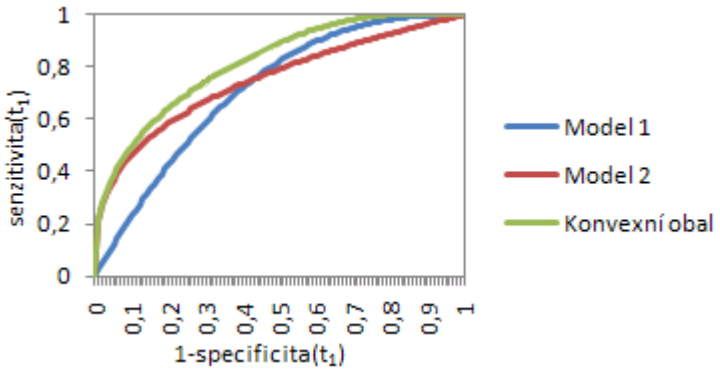
$$1 - y = 1 - Senzitivita(k) = 1 - \frac{P_{k,k}}{\sum_{i=1}^{n_t} P_{i,k}} = \frac{\sum_{i=1|i\neq k}^{n_t} P_{i,k}}{\sum_{i=1}^{n_t} P_{i,k}}, \tag{53}$$

tak po dosazení (52) a (53) do vztahu (51) vychází:

$$\frac{\left(1 - p_{klasifikace}(k)\right) c_{k,l\neq k}}{p_{klasifikace}(k) c_{l\neq k,k}} = \frac{1 - y - (1 - y')}{x' - x} = \frac{y' - y}{x' - x} \tag{54}$$

Vztah (54) definuje směrnici vývoje stejného výkonu. Všechny modely ležící na této jedné směrnici mají stejné očekávané ceny klasifikace příkladů.

Nechť existují 2 modely s křivkami ROC jako na obrázku 16. Ani jeden z modelů není optimální pro všechny strategie. Pro nalezení optimální strategie se používá konvexní obal křivek ROC. Klasifikátor, který bude ležet na tomto konvexním obalu, je optimální pro danou strategii. Lze ukázat, že pokud neleží bod křivky ROC na konvexním obalu, tak pro libovolnou rodinu linií se stejnou směrnicí lze nalézt bod, který leží na linii se stejnou směrnicí, ale dosahuje vyšších hodnot senzitivity (os y).




Obrázek 16: Konvexní obal 2 křivek ROC

### 6.2.5 T-test

T-test je statistický test, který umožňuje porovnat, v případě dvou-výběrové varianty, dvě sady čísel tak, že zjišťuje, jestli se statisticky významně od sebe liší jejich průměry. Podobně jako u $\chi^2$-testu je základ tvořen $\chi^2$ statistikou, tak u $t$-testu je základ $t$ statistika. Počítá se následovně:

$$t(x,y) = \frac{\bar{x} - \bar{y}}{S(x,y)\sqrt{\frac{1}{m} + \frac{1}{n}}} \tag{55}$$

$$\text{kde} \quad \bar{x} = \frac{1}{m}\sum_{i=1}^{m} x_i \quad \text{a} \quad \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

$$S^2(x,y) = \frac{(m-1)S_x^2 + (n-1)S_y^2}{m+n-2}$$

$$\text{kde } S_x^2 = \frac{1}{m-1}\sum_{i=1}^{m}(x_i - \bar{x})^2 \quad \text{a} \quad S_y^2 = \frac{1}{n-1}\sum_{i=1}^{n}(y_i - \bar{y})^2$$

T-test porovnává t statististiku s $(1-\alpha/2)$-kvantilem Studentova rozdělení o $(m+n-2)$ stupních volnosti $t(1-\alpha/2\,, m+n-2)$ (viz (Havránek, 1993)). Pokud platí

$$t(x,y) \geq t(1-\alpha/2\,, m+n-2) \tag{56}$$

pak $x$ je statisticky významnější než $y$.

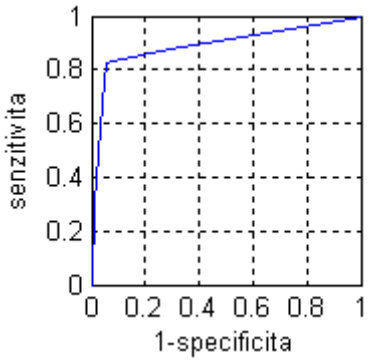
Pro porovnání dvou modelů se může vzít za $x$ a $y$ množina správností ($Acc$) nebo chyb vypočtených násobnou křížovou validací ($x$ je množina správností za jeden model a $y$ za druhý).

# 7 Systémy a jejich porovnání

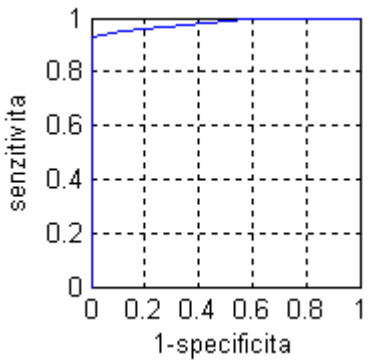
K porovnání modelů se použijí připravená data (viz kapitola příloh 10.1). Data 1 a 3 se rozdělí v poměru 4:1 a data 2a a 2b v poměru 3:1 pro všechny typy testovaných algoritmů stejně. K porovnání jednotlivých výsledků použiji obsahu pod křivkou ROC ($AUC_{ROC}$), kde platí, čím větší je obsah, tím lepší je klasifikační model.

## 7.1 ID3, C4.5 a C5 (See5)

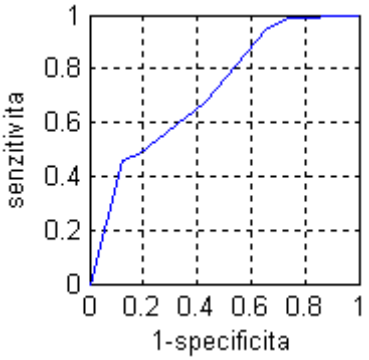
Systém využívá v algoritmu výpočtu entropie. Vytváří binární rozhodovací stromy. Má zabudovaný mechanizmus pro zjednodušení množiny odvozených pravidel. Jde o komerční program. Stáhnout se dá na http://www.rulequest.com/see5-info.html. Systém See5 nevytváří ROC křivky, proto jsou grafy vygenerované v Matlabu z rozhodovacího stromu, který systém vytvořil pro jednotlivá data.



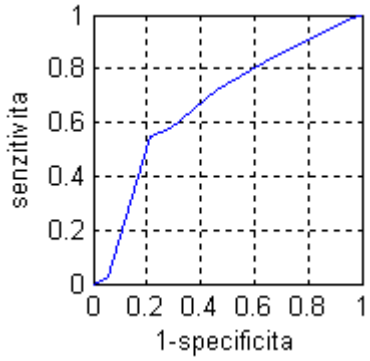

Obrázek 17: ROC vytvořeného stromu pomocí See5 pro data 1.




Obrázek 18: ROC vytvořeného stromu pomocí See5 pro data 2a.




Obrázek 19: ROC vytvořeného stromu pomocí See5 pro data 2b.



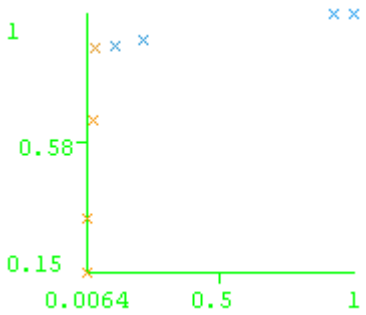

Obrázek 20: ROC vytvořeného stromu pomocí See5 pro data 3.

## 7.2 Weka

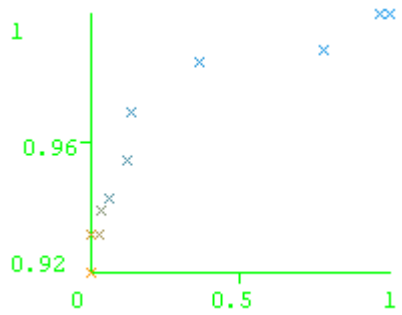
Systém obsahuje širokou škálu algoritmů, možností předzpracování a vizualizace. Je modulární a dají se k němu přidávat další moduly v Jave. Bohužel jsem nenašel postprocesing ořezávání. Například algoritmus ID3 vytvoří strom, který trénovací data zvládá bezchybně až

na jednotky (je přeučený a rozsáhlý). Jako kandidáta pro testovací data jsem si vybral klasifikační algoritmus RepTree. Jde o rychle učící se rozhodovací strom (viz http://weka.sourceforge.net/doc/weka/classifiers/trees/REPTree.html).

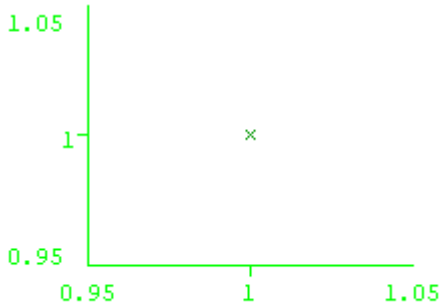
Weka se dá stáhnout z http://prdownloads.sourceforge.net/weka/weka-3-2-3.jar.



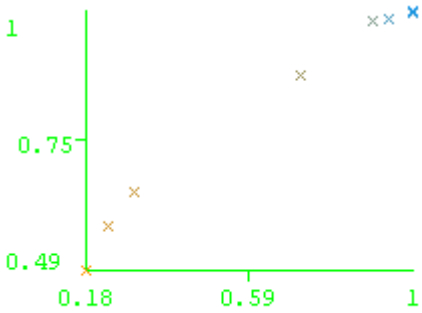

Obrázek 21: ROC vytvořeného stromu RepTree pomocí WEKA pro data 1.




Obrázek 22: ROC vytvořeného stromu RepTree pomocí WEKA pro data 2a.



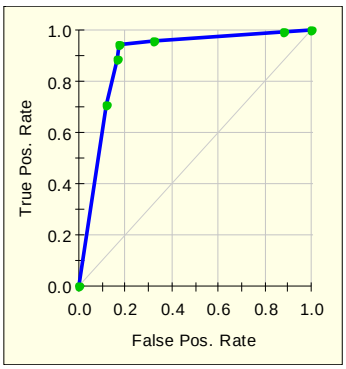

Obrázek 23: ROC vytvořeného stromu RepTree pomocí WEKA pro data 2b.



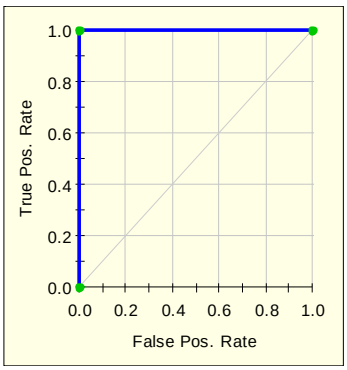

Obrázek 24: ROC vytvořeného stromu RepTree pomocí WEKA pro data 3.

### 7.3 CART

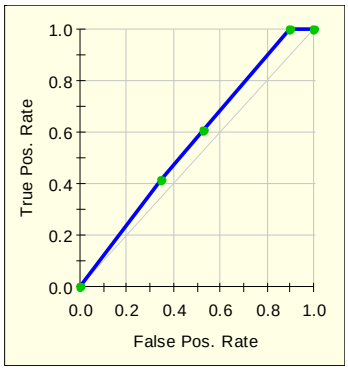
CART je zkratka od Classification and Regression Tree. Systém algoritmu je založený na výpočtu správnosti volby atributu primárně pomocí Giniho koeficientu (dá se nastavit i entropie a pár dalších). CART vytváří jenom binární stromy (tj, každý uzel má 2 větve. Umožňuje zadat matici ztrát. Většinou aplikuje prořezávání (tj. nechává vyrůst strom do maximální hloubky (tj. s velkou pravděpodobností přeučený strom) a pak ho zkracuje viz kapitola 3.1). Vytváří „Surogáty“, náhradní dělení pro případ chybějící hodnoty v atributu. Algoritmus je implementován například ve Statistice. Program se dá stáhnout ze stránek http://www.salford-systems.com/.



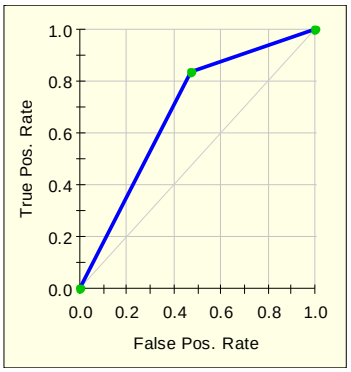

Obrázek 25: ROC vytvořeného stromu pomocí CART pro data 1.




Obrázek 26: ROC vytvořeného stromu pomocí CART pro data 2a.




Obrázek 27: ROC vytvořeného stromu pomocí CART pro data 2b.




Obrázek 28: ROC vytvořeného stromu pomocí CART pro data 3.

### 7.4 CHAID

CHAID je zkratka od Chi-squared automatic interaction detector. Patří mezi nejrozšířenější rozhodovací stromy v komerční oblasti. Jak už sám název říká, test správnosti výběru atributu se počítá pomocí $\chi^2$ statistiky. Pracuje algoritmem TDIDT. Větvení uzlu nemusí mít tolik větví, kolik je možných hodnot. Dochází k slučování statisticky málo významných hodnot. Pro všechny dvojice hodnot atributu počítá $\chi^2$ statistiku. Pokud dojde ke spojení dvou hodnot, tak tuto tabulku přepočítává. Vzhledem k tomu, že se nepočítají všechny varianty, tak algoritmus nemusí najít optimální řešení ale to za cenu velké časové úspory. Tvorbu stromu ukončuje (vytvoření listu) tehdy, pokud neexistuje statisticky významné rozdělení nebo můžou být dodatečné podmínky na počet příkladů v uzlu nebo hloubku stromu. Chybějící hodnoty CHAID považuje za další imaginární hodnotu kategorického atributu. Další informace lze nalézt také v (Soukup).

### 7.5 Výsledné porovnání jednotlivých systémů a implementovaných algoritmů

Z tabulky hodnot $AUC$ dosaženými pomocí křivek ROC za jednotlivé systémy je vidět, že žádný systém není univerzální. Každý má své lepší a horší data. Proto pro výběr systému a algoritmu je potřeba zvažovat i další věci. Například, jestli je prostředí aplikace přijatelné (aby moc nezdržovalo při práci), formát požadovaných vstupních dat a samozřejmě jestli

komerční nebo nekomerční verzi, atd. Také se není potřeba vázat jenom na jeden algoritmus, ale používat několik a nejlépe i úplně jiné metody dobývání znalostí (např. z důvodu různých druhů závislostí, které nemusí být řešitelné v daném modelu pro dobývání znalostí a nemusí být vidět – proto se vlastně používají metody dobývání znalostí, aby se na dané závislosti přišlo). V tabulce jsou i výsledky algoritmu ADTree (Alternating Decision Tree), který je ovšem kombinací rozhodovacích stromů a rozhodovacích pařezů. Jak je vidět ani tento algoritmus není dokonalý, ale v porovnání s ostatními algoritmy si vede celkově lépe.

| | **See5** | **CART** | **WEKA (RepTree)** | **WEKA (ADTree)** | **Vlastní program s $\chi^2$** | **průměr** |
|---|---|---|---|---|---|---|
| $\boldsymbol{AUC_{ROC}\ dat\ 1}$ | 0,8929 | 0,891 | 0,9284 | 0,9197 | 0,8801 | 0,90242 |
| $\boldsymbol{AUC_{ROC}\ dat\ 1.PCA}$ | 0,8130 | 0,8207 | 0,8244 | 0,8574 | 0,7736 | 0,81782 |
| $\boldsymbol{AUC_{ROC}\ dat\ 2a}$ | 0,9794 | 1 | 0,9803 | 1 | 0,9288 | 0,9777 |
| $\boldsymbol{AUC_{ROC}\ dat\ 2b}$ | 0,7244 | 0,5657 | 0,5 | 0,7917 | 0,6967 | 0,6557 |
| $\boldsymbol{AUC_{ROC}\ dat\ 3}$ | 0,6539 | 0,6831 | 0,6956 | 0,7125 | 0,7003 | 0,68908 |
| $\boldsymbol{AUC_{ROC}\ dat\ 3.PCA}$ | 0,4790 | 0,4911 | 0,5 | 0,6619 | 0,5208 | 0,53056 |
| **průměrná $\boldsymbol{AUC_{ROC}}$** | 0,7571 | 0,741933 | 0,738117 | 0,823867 | 0,75005 | |

Tabulka 4: Porovnání jednotlivých algoritmů pomocí $AUC_{ROC}$ na 4 sadách dat.

V testu byla některá data předzpracována PCA analýzou. Počet atributů nebyl snížen, aby výsledek nebyl zkreslen ztrátou informace a dal se porovnat s výsledky na původních datech. Z testů je také vidět, že pokud se data předzpracují PCA analýzou, tak výsledek nemusí být vůbec lepší, právě naopak. Pokud jsou cílové třídy závislé na nějakém atributu, který je korelován s jinými atributy, které na klasifikaci nemají skoro žádný vliv, tak po předzpracování PCA analýzou je těžší určit vliv daného atributu na výslednou klasifikaci.

# 8 Vlastní implementace rozhodovacích stromů

Program vypracovaný s touto diplomovou prací je zaměřen na vytváření rozhodovacích stromů (pokud možno v univerzálnější podobě, která by se dala časem rozšířit o další součásti). Pro testování ale bylo nutné implementovat také výpočet chyby z křivky ROC (a obsah pod křivkou), aby se dal výsledný model klasifikace porovnat s modely CART a ostatními porovnávanými v kapitole 7, které neměly žádné předzpracování vstupních dat a pak navíc s předzpracováním pomocí PCA analýzy.

Program funguje jenom v příkazové řádce. Výstupem programu je vytvořený rozhodovací strom podle zvolených parametrů, hodnota chyby, export souřadnic pro zlomové body ROC křivky, z kterých se dá tato křivka pro danou třídu vykreslit buď pomocí MS Excel, v MATLABu nebo jiném programu.

Při vývoji byla použita knihovna boost 1.39 (www.boost.org), která je potřebná pro případné kompilace programu.

### 8.1.1 Parametry programu v příkazovém řádku

Program pracuje jenom v příkazovém řádku. Parametry programu:

-d ‘soubor’ vstupní soubor trénovacích příkladů ve formátu souborů See5 („-“ default).
-n ‘soubor’ vstupní soubor jmen a omezení atributů včetně informace o atributu s třídami klasifikace ve formátu See5.
-t ‘soubor’ vstupní soubor testovacích dat ve formátu See5.
-c ‘soubor’ vstupní soubor cen za špatnou klasifikaci ve formátu See5 (prozatím nefunkční).
-o ‘soubor’ výstupní soubor, do kterého bude uložen výsledek („-“ default).

Jeden ze vstupních souborů a jeden z výstupních souborů může být roven „-“ a znamenat standardní vstup/výstup.

-T parametr=hodnota[;parametr=hodnota[…]]
nastavení jednotlivých parametrů vytváření rozhodovacího stromu:
‘spravnost’ prozatím jenom “ChiKvadrat” (default a prozatím se nedá měnit).
‘chyba’ prozatím jenom “AUC_ROC” (default a prozatím se nedá měnit).
‘prorez’ “true” pro povolení prořezávání,
“false” pro zakázání prořezávání (default).
‘minPocVListu’ minimální počet trénovacích příkladů v jednom listě (defaultně 10).
‘minPocVUzlu’ minimální počet trénovacích příkladů v jenom uzlu (defaultně 10).
‘skoncPokudNezavisle’

„true“ pro možné ukončení dělení uzlu, pokud jsou všechny atributy statisticky nezávislé na klasifikačních třídách. „false“ pro pokračování i po výpočtu statistické nezávislosti atributů na třídách. (prozatím nefunkční).

‘prorezAlg’ algoritmus zpětného prořezávání (prozatím nefunkční).

‘procZastoupeniPrikladu’

procento zastoupeni příkladů na danou třídu, pro kterou se už nemusí provádět dělení. Číslo mezi 0 až 1. (defaulně 0.75)

‘algoritmus’ typ algoritmu (prozatím nefunkční).

-O parametr[=hodnota][;parametr[=hodnota][…]]

nastavení jednotlivých parametrů pro vypsání výsledků z vytvoření a testování rozhodovacího stromu „true“ nebo bez nastavení znamená, že se daná položka vypíše, „false“, že se položka nevypíše:

‘strom’ textová podoba rozhodovacího stromu.

‘spravnosti’ do vypisovaného stromu vypisuje i hodnotu správnosti výběru daného atributu s daným rozdělením hodnot do větví.

‘matzam’ matice záměn.

‘AUC_ROC’ tabulka chyb AUC_ROC pro jednotlivé třídy a celkový součet.

‘ROC’ tabulka x-ových a y-ových souřadnic zlomových bodů křivky ROC pro jednotlivé třídy (default = „false“).

-h vypsání nápovědy.

### 8.1.2 Rozhodovací strom

Vstupem pro rozhodovací strom jsou data, která obsahují jenom celá čísla (kategoriální atributy), přirozená čísla (numerické atributy) nebo hodnotu NaN (pro nedefinovanou hodnotu atributu). Na tuto podobu dat se vstupní data vnitřně sama převedou. Algoritmus výpočtu je od datové struktury separovaný, aby bylo možné časem přidávat další modely výpočtu jenom formou zásuvných modulů (momentálně je přímo (ne formou modulu) implementován jenom algoritmus s výpočtem $\chi^2$-statistky).

Algoritmus pro tvorbu rozhodovacího stromu má následující vlastnosti (kromě obmezení, které se dají nastavit) pro hledání reprezentanta nového podstromu:

- Pro numerické atributy nalézá hraniční bod, kterým rozdělí hodnoty do 2 skupin.
- Pro kategorické atributy dochází k plnému větvení uzlu pro všechny možné hodnoty. V případě, kdy by nějaká vzniklá větev měla být reprezentována menším počtem trénovacích příkladů, než je povoleno pro uzel nebo list, tak dojde k jejímu sloučení s jinou větví a to tak, aby spočtená správnost byla co největší.

Rozhodovací strom byl implementován jako datová objekt, který si interně udržuje:

- pole uzlů (první položka je vždy kořen),
- pole odkazů na všechny listy stromu,
- pole názvů jednotlivých atributů s povolenými hodnotami,
- odkaz na atribut představující třídy,
- matici cen za chybnou klasifikaci (nepoužito),
- typ výpočtu správnosti výběru uzlu,
- nastavení pro prořezávání:
    - povolení prořezávání,
    - minimální počet příkladů v listu,
    - minimální počet příkladů v rodičovském uzlu,
    - percentuální hodnota zastoupení nejpočetnější třídy,
    - informaci, zda skončit v dělení pokud se statisticky určí, že jsou všechny atributy nezávislé na daných třídách,
- typ počítané chyby,
- celková nejlepší spočtená chyba pro trénovací data,
- celková spočtená chyba pro posledně testovaná data,
- celkový počet kroků prořezávání (nepoužito),
- optimální krok prořezání (s nejmenší chybou) (nepoužito).

Každý řádek pole uzlů obsahuje:

- odkaz na rodičovský uzel (řádek),
- odkaz na svého bratra zleva (řádek),
- odkaz na svého bratra zprava (řádek),
- odkaz na svého prvního syna (řádek),
- počet synů,
- počet trénovacích příkladů za jednotlivé třídy (pole),
- počet posledně testovaných příkladů za jednotlivé třídy (pole),
- atribut, který uzel reprezentuje,
- hodnotu atributu reprezentovanou rodičovským uzlem, tj. větev, kterou je uzel propojen s rodičovským uzlem (je pro kategoriální atributy rovno bitovému poli, které říká, pro které hodnoty je určen a pro numerické atributy je 2x double hodnotě - levé a pravé hranici intervalu),
- hodnotu správnosti výběru daného atributu,
- hloubku prořezání (číslo určující pořadí, v kterém byl daný uzel uříznut při zpětném prořezávání) (nepoužito).

Ne všechny položky jsou v programu využívány, některé slouží spíše k budoucí doimplementaci dalších funkčností.

V programu existuje obecný rozhodovací strom (class CRozhodovaciStrom), který je jenom obecným objektem, nad kterým je možné postavit případně dokonalejší variantu. Tento

objekt má rozhraní, pomocí kterého můžou algoritmy s tímto stromem pracovat a vytvářet ho.

Trénovací a testovací data se natahují do paměti celé (ale postupně).

### 8.1.3 Implementace výpočtu křivky ROC a $AUC_{ROC}$

V této kapitole je popsán algoritmus výpočtu křivky ROC a hodnoty $AUC_{ROC}$.

Pro vytvoření křivky ROC se nemusí stále zvyšovat práh po malých částech, přepočítat matici záměn a spočítat specificitu a senzitivitu. Stačí na to jednoduchý algoritmus, který pracuje s relativními četnostmi zařazených testovacích a trénovacích příkladů do jednotlivých listů:

#### Algoritmus 4: Výpočtu zlomových bodů ROC křivky pro třídu $k$

1. Nastavme:
   a) $Tr_l = 1 - \frac{p_l(k)}{\sum_{i=1}^{n_t} p_l(i)}$,
      kde $p_l(i)$ je počet **trénovacích** příkladů třídy $i$ zařazených rozhodovacím stromem do listu $l$.
      $n_t$ je počet všech klasifikačních tříd.
      $Tr_l$ je četnost **trénovacích** příkladů, které nepatří do třídy $k$ a jsou rozhodovacím stromem zařazeny do listu $l$.
   b) $q_l(i)$ je počet **testovacích** příkladů třídy $i$ zařazených rozhodovacím stromem do listu $l$.
   c) Práh = 0.
   d) *xy* = {[0,0]} (pole bodů zlomu křivky ROC).
2. Pokud není Práh rovný 1, tak opakuj kroky 2a až 2b.
   2a. Nastav práh na nejmenší hodnotu $Tr_l$, která je větší než dosavadní hodnota prahu.
   2b. Spočti hodnoty senzitivity a specificity (viz vzorce (47) a (52)) a přidej nový bod zlomu do křivky (do *xy*), kde čitatel pro senzitivitu a (1-specificitu) se spočítá:
   $P_{k,k} = \sum_{(\forall l | Tr_l \leq práh)} q_l(k)$ ,
   $\sum_{(\mathrm{i}=1|i\neq k)}^{\mathrm{n_t}} \mathrm{P_{k,i}} = \sum_{(\forall l | Tr_l \leq práh)} \sum_{i=1|i\neq k}^{n_t} q_l(i)$.

V tomto algoritmu lze jmenovatele senzitivity a specificity spočítat na začátku, protože jsou celou dobu stejné. Čitatele stačí jenom aktualizovat o hodnoty z listů, které právě překročily hodnotu prahu, a už se nemusí udržovat informace o těchto listech, protože se jejich přínos nezmění pro žádnou větší hodnotu prahu.

Proč algoritmus funguje je vysvětleno níže.

Nechť existuje rozhodovací strom s $n$ listy v nichž je četnost trénovacích příkladů třídy $j$ v listu i rovna $p_i(j)$. Nechť je posloupnost $\{p_i(k)\}_i$ pro $i \in \{1,2,\dots,n\}$ neklesající. Pak pro práh $\theta \in \langle p_c(k), p_{c+1}(k)\rangle$ platí, že se nemění $senzitivita(k)$ (tj. y-ová souřadnice křivky ROC pro třídu $k$ je pořád stejná), jelikož senzitivita se změní jenom tehdy, pokud se

změní četnost správné klasifikace příkladu do třídy $k$. Tato změna může nastat jenom v případě, že práh $\theta$ překročí jednu z pravděpodobností $p_i(k)$, což z předpokladu o neklesající posloupnosti $\{p_i(k)\}_i$ a zvoleného intervalu pro $\theta$ není možné. (Příklad, který patří do třídy $k$ a byl rozhodovacím stromem klasifikován do třídy $l$ je započítán do $P_{k,l}$. Změna klasifikace příkladu z třídy $k$ původně klasifikovaného do třídy $l \neq k$ a nově do třídy $m \neq k$ je nezajímavá, protože ve vzorci senzitivity (47) dochází k celkovému součtu všech příkladů z třídy $k$.)

To samé lze dokázat pro $(1 - specificitu)$ tedy pro souřadnici na ose x. Takže pro výpočet bodů, kterými prochází křivka ROC pro třídu $k$, stačí uchovávat v listech jednotlivé četnosti trénovacích a testovacích příkladů. Tento algoritmus se dá využít i u jiných modelů klasifikace, které fungují podobně jako rozhodovací stromy (například rozhodovací pravidla). Obsah pod křivkou ROC ($AUC_{ROC}$) se poté spočítá jako součet obsahů n+1 lichoběžníků, kde $n$ je počet listů rozhodovacího stromu.

# 9 Závěr

Závěrem zrekapituluji hlavní výsledky práce, kterých bylo dosaženo.

V práci bylo zrekapitulováno vytváření rozhodovacích stromů, různé změny a jejich důsledky. Dále byl podán přehled různých vizualizací, jejich výhody a nevýhody. Bylo provedeno experimentální porovnávání různých modelů s různými způsoby ukončovacích podmínek. Bylo ukázáno, že každá z variant má v některých případech navrch a v některých zase ne. To znamená, že v případě úplně neznámých dat se nedá určit, která metoda je lepší. V případě potřeby se musí vyzkoušet všechny. Nebo použít jiné modely dobývání znalostí popřípadě jejich kombinace.

Z podaných znalostí o algoritmech spojenými s rozhodovacími stromy se dá dojít k závěru, že rozhodovací stromy jsou vhodné pro případy, kdy:

1) Příklady se dají relativně lehce reprezentovat hodnotami atributů (například pro nestrukturalizovaná data je lepší použít úplně jinou metodu z dobývání znalostí).
2) Cílem je klasifikovat příklady do malého počtu tříd (v případě velkého počtu tříd se už špatně provádí kontrola, jestli je klasifikace dostačující, a v případě nekonečného počtu tříd byl rozhodovací strom teoreticky nekonečný, nebo by do nějakých tříd prostě nedokázal klasifikovat). V případě velkého počtu tříd je lepší úlohu rozložit na několik menších.
3) Hledaný popis řešení se dá vyjádřit konečným počtem disjunkcí (disjunkce jednotlivých pravidel tvořených konjunkcemi jednoduchých porovnání). V případě, že je tento počet nekonečný, jako například v případě zbytku po dělení celých čísel, dochází při vytváření rozhodovacích stromů buď k přeučení, nebo jenom omezené schopnosti klasifikovat příklady z určité podobné množiny příkladů, z které byla trénovací data. Na druhou stranu špatnou schopností klasifikovat daná data různými modely rozhodovacích stromů lze také získat nějakou informaci o daných datech.
4) Data mohou být zatížená šumem (pozor ale na normování například při PCA analýze, kdy šum může způsobit velké nepříjemnosti při zmenšování počtu atributů).

Z testů při použití PCA analýzy jako prostředku k předzpracování dat je vidět, že výsledky jsou o něco málo horší než bez předzpracování. Navíc je nutné vzít v úvahu, že se hůře zpětně získávají znalosti pro původní atributy. Proto se dá říci, že nástroje na úpravu dat na základě korelací jednotlivých atributů jsou určeny opravdu pouze k možnému zmenšení velkého počtu atributů. Lepší použití těchto metod je dle mého názoru jenom na atributy, které nesplňují minimální korelace s klasifikačními třídami, popřípadě i odděleně jenom na atributy, které tuto korelaci splňují.

Možné navázání na tuto práci vidím například v nalezení obecnější varianty ROC křivky pro více atributů, která vyřeší problém, jak v takovém případě pracovat s prahem (práh nemusí být jenom jedna hodnota, ale celý vektor s nějakými vlastnostmi) včetně porovnání s podobnými metodami popřípadě jenom porovnání podobných křivek pro dvě třídy. Jiným

pokračováním by mohlo být využití evolučních algoritmů jako způsobu učení (nebo jeho doplňku) pro vytváření rozhodovacích stromů. Zajímavé by bylo i hledání původních závislostí v datech po předzpracování neuronovými sítěmi popřípadě PCA analýzou. Podobné projekty už myslím byly rozběhnuty a existují již částečné výsledky.

# 10 Přílohy

## 10.1 Data pro testování

1. **Náhodně vygenerovaná**

   Umělá data obsahují 1000 vygenerovaných příkladů s 10 atributy a následujícími vlastnostmi:

   1.-3. atribut je kategorický s 3 hodnotami vygenerovanými náhodně (hodnoty 0 - 2).
   4.-6. atribut obsahuje náhodné reálné číslo z intervalu $\langle -1,1 \rangle$.
   7. atribut je náhodné reálné číslo z intervalu $\langle -1, 0 \rangle$.
   8. atribut je kategorický s 4 hodnotami vygenerovaný náhodně (hodnoty 0 - 3).
   9. atribut je náhodné reálné číslo z intervalu $\langle 0, 1 \rangle$.
   10. atribut je kategorický s 2 hodnotami vygenerovaný náhodně (hodnoty 0 a 1).

   Třídy klasifikace: $\begin{cases} 0, & (a_1 < a_2) \wedge (a_4 < 0) \\ 1, & jinak \end{cases}$ s šumem 10%.

2. **EuroMISE Discovery Challenge 2003**

   Jde o lékařská data. Pro testy bude brána jenom tabulka Entry, která obsahuje záznamy od 1417 lidí. Vysvětlivky k jednotlivým sloupcům jsou uvedeny na adrese http://euromise.vse.cz/challenge2003/data/entry/index.php?page=uvod. Jako třída klasifikace se vezme, zda je nebo není přítomno riziko:

   a) kouření „KOURRISK“ (ano: 620, ne: 796, neurčeno: 1)
   b) pozitivní rodinná anamnéza „RARISK“ (ano: 1158, ne: 250, neurčeno: 9)

   Studie (STULONG) byla realizována na 2. katedře medicíny, 1. Lékařské fakulty Univerzity Karlovi a Karlovou universitní nemocnicí, U nemocnice 2, Praha 2 (vedoucí Prof. M. Aschermann, MD, SDr, FESC), pod dohledem Prof. F. Boudík, MD, ScD, se spoluprácí s M. Tomečková, MD, PhD a Ass. Prof. J. Bultas, MD, PhD. Data byla převedena do elektronické podoby Europským Centrem Medicínské Informatiky, Statistikou a Epidemiologii Karlovy University a Akademii věd (vedoucí Prof. RNDr. J. Zvárová, DrSc). Do nynějška je analýza dat podporována grantem Ministerstva Vzdělávání ČR č. LN 00B 107.

3. **Data nejmenované německé banky**

   1. Data obsahují 1000 příkladů (klientů), kterým byl přidělen úvěr (zdroj: www.stat.uni-muenchen.de).
   2. Třídy klasifikace jsou, zda klient má nebo nemá dostat úvěr. V datech je informace, zda klient úvěr v pořádku splatil (DEFAULT=0), či nikoli (DEFAULT=1).

3. Atributy:
    - **DEFAULT** – selhání klienta (viz bod 2)
    - **DOBA** – doba splatnosti úvěru v měsících
    - **VÝŠE** – výše půjčky
    - **ÚČEL** – účel použití úvěru (1=nové auto, 2=starší auto, 3=zařízení bytu, 4=domácí spotřebiče, 5=oprava, 6=vzdělání, rekvalifikace, 7=ostatní)
    - **GARANCE** – existence ručitelů (0=úvěr nezajištěn ručiteli, 1=úvěr zajištěn ručiteli)
    - **KONTO** – výše zůstatku na běžném účtu (1=nemá běžný účet, 2=méně nebo rovno 0 DM, 3=0 DM až 200 DM, 4=vice než 200 DM)
    - **ÚSPORY** – výše úspor
    - **AKTIVA** – nejcennější vlastněné aktivum (1=žádná aktiva, 2=osobní auto, 3=životní pojištění, 4=dům, pozemek)
    - **ÚVĚRY** – souběžné vlastnictví dalších úvěrů (0=klient nemá další úvěry, 1= klient vlastní další úvěry)
    - **ÚVĚRYZDE** – počet dřívějších půjček v této bance (1=1, 2=2 nebo 3, 3=4 nebo 5, 4=6 a více)
    - **MORÁLKA** – platební kázeň u dřívějších půjček (0=předchozí úvěry řádně splaceny, 1=bezproblémové splácení současných úvěrů, 2=žádné předchozí úvěry, 3=problémy s běžným účtem, 4=váhavé splácení předchozích úvěrů)
    - **PODÍL** – splátka v % z disponibilního příjmu (1=méně než 20%, 2=20% až 25%, 3=25% až 35%, 4=více než 35%)
    - **VĚK** – věk žadatele v letech
    - **STAV** – rodinný stav (1=muž - svobodný, rozvedený, vdovec, 2=žena - svobodná, rozvedená, vdova, 3=muž – ženatý, 4=žena – vdaná)
    - **ZAM** – zaměstnání (1=nezaměstnán, 2=nekvalifikovaný pracovník, 3=kvalifikovaný pracovník, 4=vedoucí pracovník)
    - **DOBAZAM** – počet let v současném zaměstnání (1=méně než 1 rok, 2=1 až 4 roky, 3=4 až 7 let, 4=vice než 7 let)
    - **DOBABYD** – počet let v současném bydlišti (viz DOBAZAM)
    - **CIZINEC** – žadatel (0=rezidentní klient, 1=klient je cizinec)

Zbylé atributy se nebudou brát v úvahu.

## 10.2 Obsah přiloženého média

| relativní cesta | Obsah |
|---|---|
| Data | Obsahuje data použita při testování. Každý soubor má předponu s číslem skupiny dat (viz kapitola 10.1). Existuje více variant z důvodu jiných vstupů pro jednotlivé programy. |
| Vysledky | Adresář obsahuje výsledky z testování, uložené obrázky stromů, ROC křivek a další dostupné informace v daném programu. Výsledky jsou rozděleny do adresářů, jejichž jméno je název programu, z kterého vyšly. |
| zdrojoveKody | Adresář obsahuje všechny zdrojové kódy, které byly pro tuto diplomovou práci mnou vytvořené (MATLAB, C++). |
| Rozhodovaci stromy.pdf | Tato diplomová práce. |

# 11 Reference


[1] Anděl Jiří: Základy matematické statistiky [Kniha]. - Praha : Preprint, 2002.

[2] Bečvář J.: Lineární algebra [Kniha]. - Praha : Matfyzpress, 2000.

[3] Berka Petr: Dobývání znalostí z databází [Kniha]. - Praha : Academia, 2003.

[4] Betinec Martin a Prchal Luboš: Poznámky k ROC křivkám [Online]. - 7 2009. - http://www.karlin.mff.cuni.cz/~antoch/robust06/postery/betinec.pdf.

[5] Havránek T.: Statistika pro biologické a lékařské vědy [Kniha]. - Praha : Academia, 1993.

[6] Kung S. Y.: Digital neural networks [Kniha]. - New Jersey : PTR Prentice Hall, Englewood Cliffs, 1993.

[7] Louis A. K., Maas P. a Rieder A. Wavelets: Theory and applications [Kniha]. - [místo neznámé] : Wiley, 1997.

[8] M. Berry G. Linoff: Data Mining Techniques For Marketing, Sales and Customer Support [Kniha]. - [místo neznámé] : John Wiley & Sons, 1997.

[9] Mařík Vladmír a další: Umělá inteligence (4) [Kniha]. - Praha : Academia, 2003.

[10] Michalski R. S.: A planer geometric model for presenting multidimensional discrete spaces and multiple-valued logic functions [Kniha]. - Universite of Illinois : Tech. Rep., 1978.

[11] Militký Jiří a Meloun Milan: Metoda hlavních komponent a explorativní analýza [Online]. - 7 2009. - http://meloun.upce.cz/docs/publication/127a.pdf.

[12] Myslík Vladimír: Neuronové sítě [Online]. - 1996. - 6 2009. - http://aldebaran.feld.cvut.cz/~xmyslik/www/neural.html.

[13] neznámí: Počítačová podpora lékařského rozhodování [Online]. - 2009. - http://biof.lf1.cuni.cz/mkuba/12-INF2008-2009-Rozhodovani.ppt.

[14] neznámí: Receiver operating characteristic [Online]. - 7 2009. - http://en.wikipedia.org/wiki/Receiver_operating_characteristic.

[15] neznámí: Rozhodovací stromy a jejich konstrukce z dat [Online]. - 6 2009. - http://datel.felk.cvut.cz/xui1/2007ZS/UI6-RozhodovaciStromy.ppt.

[16] neznámí: Vlastní čísla [Online]. - 7 2009. - flow.kmo.tul.cz/~www/czech/vyuka/lag/vlcisla.pdf.

[17] Rojas R.: Neural Networks: A Systematic Introduction [Kniha]. - [místo neznámé] : Springer-Verlag, 1996.

[18] Soukup Jan: Rozhodovací stromy (lekce 9) [Online]. - 6 2009. - http://samba.fsv.cuni.cz/~soukup/spousta/lekce/09Stromy.ppt.