# NEBINÁRNÍ KLASIFIKAČNÍ STROMY

STANISLAV KEPRTA

MFF UK, Praha

ABSTRACT. Tato práce se zabývá klasifikačními stromy. Většina dosud navržených algoritmů konstruuje striktně binární stromy. Ale vícecestná štěpení mohou někdy lépe vystihnout strukturu dat než binární. Hlavním cílem článku je navrhnout metody pro konstrukci a porovnávání vícecestných štěpení.

## 1. ÚVOD

V praxi často potřebujeme řešit klasifikační problém – předpovídat, do které třídy objekt patří, když známe výsledky nějakých měření (prediktory) na tomto objektu. V této práci se budeme zabývat situací, kdy máme k dispozici soubor obsahující naměřené hodnoty na několika objektech a známe, do které třídy tyto objekty ve skutečnosti náleží. Na základě tohoto souboru chceme prozkoumat závislosti mezi naměřenými veličinami a zařazením do tříd a naučit se co nejpřesněji tato zařazení předpovídat. Velmi populárním prostředkem jsou klasifikační stromy, a to pro svoji jednoduchost, srozumitelnost, schopnost zpracovávat různé typy prediktorů a různé závislosti mezi prediktory navzájem i mezi prediktory a třídami.

Bylo navrženo mnoho různých metod konstrukce klasifikačních stromů. Zmiňme se alespoň o FIRMu navrženém Hawkinsem v [6], Ciampiho algoritmu [3] a CARTu popsaném v monografii [1]. Více citací lze nalézt v [10].

Cílem této práce je zobecnit algoritmus CART (Classification And Regression Trees) konstruující binární klasifikační stromy a jeho vylepšení popsané Gelfandem *et al* v [5] na konstrukci i nebinárních stromů, které často poskytují snáze interpretovatelné a přesnější výsledky než stromy striktně binární.

V kapitole 2 jsou definovány základní pojmy, které budeme dále potřebovat. Kapitola 3 popisuje metodologii konstrukce klasifikačních stromů. V kapitole 4 jsou navrženy a podrobně popsány algoritmy konstrukce a porovnávání vícecestných štěpení. V kapitole 5 je uveden ilustrativní příklad.

## 2. ZÁKLADNÍ POJMY A DEFINICE

Nechť $(X', Y)'$ je náhodný vektor, $X$ nabývající hodnot z $\mathcal{X} \subset \mathbb{R}^M$ a $Y$ z $\mathcal{C} = \{1, 2, \ldots, J\}$. $X$ se nazývá vektor prediktorů, $Y$ je odpovídající třída. Naším cílem je odhadovat $Y$ na základě pozorovaných vysvětlujících náhodných veličin ve vektoru $X = (X_1, X_2, \ldots, X_M)'$.

Autor děkuje RNDr. Jaromíru Antochovi, CSc., za všestrannou pomoc při práci na tomto problému a při přípravě tohoto článku.

Předpokládejme, že máme soubor dat $\mathcal{L} = \{(x'_n, y_n)', \ n = 1, \ldots, N\}$, který byl získán jako realizace náhodného výběru. Především chceme zkonstruovat *klasifikační pravidlo*.

**Definice 2.1.** Klasifikační pravidlo (klasifikátor) je funkce $d(x)$ definovaná na $\mathfrak{X}$ taková, že každému $x \in \mathfrak{X}$ přiřazuje právě jedno číslo z $\mathcal{C}$.

**Poznámka 2.1.** To znamená, že $d(x)$ určuje disjunktní rozklad prostoru $\mathfrak{X}$ na podmnožiny $\mathcal{A}_1, \ldots, \mathcal{A}_J$ takový, že

1) $\bigcup_{j=1}^{J} \mathcal{A}_j = \mathfrak{X}$;
2) $\mathcal{A}_i \cap \mathcal{A}_j = \emptyset, \ 1 \leqq i \neq j \leqq J$;
3) $\forall x \in \mathfrak{X} \ \ d(x) = j \iff x \in \mathcal{A}_j$.

*Klasifikační strom* je klasifikační pravidlo získané rekurzivním rozkladem podmnožin prostoru $\mathfrak{X}$ do dvou nebo více podmnožin, které odpovídají „uzlům" stromu. Při rozkladu přitom začínáme celým prostorem $\mathfrak{X}$, který odpovídá „kořenu" stromu. Připomeňme si základní terminologii z teorie grafů, která se týká stromů a zaveďme některá označení.

**Definice 2.2.** *Stromem* rozumíme konečnou neprázdnou množinu $T$ přirozených čísel a funkci *sons*, která každému $t \in T$ přiřazuje podmnožinu $T$. Označíme symbolem $|T|$ počet prvků množiny $T$. Na funkci *sons* klademe následující požadavky:

1) $\forall t \in T$ buď $sons(t) = \emptyset$, nebo $|sons(t)| \geqq 2$;
2) $\forall t \in T \ s \in sons(t) \Longrightarrow s > t$;
3) $\forall s \in T$ různé od nejmenšího prvku v $T$, $\exists!\ t \in T$ takové, že $s \in sons(t)$.

**Terminologie.**

- Každé $t \in T$ se nazývá *uzel* stromu.
- Nejmenší prvek $T$ se nazývá *kořen* stromu a značí se $root(T)$.
- Takové $t \in T$, že $sons(t) = \emptyset$, se nazývá *koncový uzel* nebo *list*.
- Uzel $s$ se nazývá *syn* uzlu $t$ a uzel $t$ se nazývá *otec* uzlu $s$, jestliže $s \in sons(t)$.
- Podle definice 2.2 víme, že ke každému uzlu kromě kořenu existuje právě jeden otec. Proto můžeme zavést funkci $father(t)$, která každému $t \in T - \{root(T)\}$ přiřazuje jeho otce.
- Pokud $t \in T$ lze zapsat jako $t = father(s)$, nebo jako $t = father(father(s))$, nebo ... , nazývá se $t$ *předchůdcem* $s$ a $s$ *následníkem* $t$.

Mějme neprázdnou podmnožinu $T_1 \subset T$. Definujeme funkci $sons_1(\cdot)$ předpisem

$$\forall t \in T_1 \ \ sons_1(t) = sons(t) \cap T_1.$$

**Definice 2.3.** Pokud $T_1$ a $sons_1(t)$ vyhovují definici stromu, říkáme, že $T_1$ je *podstrom* stromu $T$. $T_1$ se nazývá *prořezaný podstrom* $T$ (s označením $T_1 \preccurlyeq T$, případně $T_1 \prec T$ místo $T_1 \preccurlyeq T$ a $T_1 \neq T$), jestliže je $T_1$ podstrom $T$ a $root(T) = root(T_1)$.

**Terminologie.**

- *Větví* stromu $T$ vycházející z $t \in T$ (označení $T_t$) rozumíme podstrom stromu $T$, který obsahuje $t$ jako kořen a všechny následníky $t$.
- *Prořezáním stromu* $T$ rozumíme činnost, která ze stromu $T$ vytvoří prořezaný podstrom $T_1 \preccurlyeq T$ uříznutím některých větví. Uříznutí větve $T_t$ spočívá v tom, že

jsou odstraněni všichni následníci $t$ a $t$ se stane koncovým uzlem. Strom vzniklý z $T$ uříznutím větve $T_t$ značíme $T - T_t$.

Označme ještě $\tilde{T}$ množinu listů stromu $T$. Nyní můžeme spojit definice stromu a klasifikačního pravidla.

**Definice 2.4.** *Klasifikační strom* je strom $T$, kde každému $t \in T$ přiřadíme podmnožinu $\mathcal{A}(t) \subset \mathfrak{X}$ a třídu $j(t) \in C$. Přitom požadujeme, aby

1) $\{\mathcal{A}(t);\ t \in \tilde{T}\}$ byl disjunktní rozklad $\mathfrak{X}$;
2) $\forall t \in T - \tilde{T}$ $\{\mathcal{A}(s);\ s \in sons(t)\}$ byl disjunktní rozklad $\mathcal{A}(t)$.

Klasifikační pravidlo odpovídající takovémuto klasifikačnímu stromu potom zní:

$$\forall \boldsymbol{x} \in \mathfrak{X}\ \ \forall t \in \tilde{T}\ \ d(\boldsymbol{x}) = j(t) \iff \boldsymbol{x} \in \mathcal{A}(t).$$

## 3. ITERAČNÍ RŮST A PROŘEZÁVÁNÍ STROMU

### 3.1 Metody konstrukce klasifikačních stromů

Bylo navrženo několik různých metodologií konstrukce klasifikačních stromů. Zde stručně popíšeme jednu z nich, založenou na [1]. Strom je konstruován rekurzivním rozkladem prostoru $\mathfrak{X}$. To znamená, že nejprve je kořenu 1 přiřazen celý prostor $\mathfrak{X}$, tj. $\mathcal{A}(1) = \mathfrak{X}$. Potom musíme nalézt vhodné štěpení. Štěpení je určeno podmínkami kladenými na $x_1, \ldots, x_M$. Tyto podmínky určují rozklad prostoru $\mathfrak{X} = \mathcal{A}(1) = \bigcup_{s \in sons(1)} \mathcal{A}(s)$. Poté je stejná metoda použita rekurzivně na každou množinu $\mathcal{A}(s), s \in sons(1)$, atd.

Konstrukce klasifikačního stromu se tak skládá ze tří kroků:

1) Výběr štěpícího pravidla v každém uzlu.
2) Rozhodnutí, kdy je uzel koncový.
3) Přiřazení třídy vysvětlované proměnné každému koncovému uzlu.

Předpokládejme, že konstruujeme strom $T$ na základě souboru dat $\mathcal{L}$. V každém uzlu $t \in T$ označme $p(t) = N(t)/N$, kde $N(t)$ je počet dat v $\mathcal{L}$ takových, že $\boldsymbol{x}_n \in \mathcal{A}(t)$; $p(t)$ je tedy odhad $P(\boldsymbol{X} \in \mathcal{A}(t))$ založený na $\mathcal{L}$. Dále označme $p(j|t) = N_j(t)/N(t)$, $j = 1, \ldots, J$, kde $N_j(t)$ je počet dat v $\mathcal{L}$ takových, že $\boldsymbol{x}_n \in \mathcal{A}(t)$ a $y_n = j$; $p(j|t)$ je tedy odhad $P(Y = j | \boldsymbol{X} \in \mathcal{A}(t))$ založený na $\mathcal{L}$.

**ad 3)**

Nejjednodušší je krok 3) — přiřazení třídy $j(t)$ uzlu $t$. Pro jakoukoliv volbu $j(t)$ dostaneme odhad chyby klasifikace $1 - p(j(t)|t)$. Tento odhad je minimální, volíme-li $j(t)$ tak, aby

$$p(j(t)|t) = \max_j p(j|t) = \max_j \frac{N_j(t)}{N(t)}.$$

Chybu klasifikace stromu $T$ definujeme předpisem

$$R(T) = \sum_{t \in \tilde{T}} R(t),$$

kde $R(t) = M(t)/N$ a $M(t)$ je počet dat v $\mathcal{L}$, pro která $\boldsymbol{x}_n \in \mathcal{A}(t)$ a $y_n \neq j(t)$. Tak dostaneme

$$R(t) = [1 - p(j(t)|t)]\, p(t) = \left[1 - \max_j p(j|t)\right] p(t).$$

ad 1)

Nyní se podíváme na problém výběru štěpení $s = s(t)$ ve vnitřním uzlu $t \in T - \tilde{T}$. Definujeme *míru nečistoty* $i(t)$ uzlu $t$ jako

$$i(t) = \Phi\left(p(1|t), p(2|t), \ldots, p(J|t)\right),$$

kde $\Phi$ je funkce na kterou klademe tyto požadavky:

1) $\Phi = \Phi(p_1, \ldots, p_J)$ je definována pro taková $p_1, p_2, \ldots, p_J$, pro která $\sum_{j=1}^{J} p_j = 1$ a $0 \leqq p_j \leqq 1$, $j = 1, \ldots, J$.
2) $\Phi \geqq 0$.
3) $\Phi$ nabývá maxima pro $p_1 = p_2 = \cdots = p_J = 1/J$.
4) $\Phi$ nabývá minima, je-li $p_j = 1$ pro nějaké $j$.
5) $\Phi$ je ryze konkávní.

To znamená, že nečistota uzlu je největší, pokud jsou všechny třídy zastoupeny rovnoměrně, a nejmenší, pokud uzel obsahuje data pouze z jedné třídy.

*Pokles nečistoty* při štěpení uzlu $t$ pomocí štěpení $s$ definujeme jako

$$\Delta i(s, t) = i(t) - \sum_{s \in sons(t)} i(s) \frac{p(s)}{p(t)},$$

kde $sons(t)$ odpovídá štěpení $s$. Označme $\mathcal{S}_k$ množinu uvažovaných štěpení na $k$ podmnožin. Štěpení $s_k(t) \in \mathcal{S}_k$ považujeme za nejlepší v $\mathcal{S}_k$, jestliže

$$\Delta i(s_k(t), t) = \max_{s \in \mathcal{S}_k} \Delta i(s, t).$$

Množinu štěpení $\mathcal{S}$ většinou volíme jako množinu rozkladů podle hodnot jednotlivých prediktorů. Přitom uvažujeme několik typů prediktorů.

1) Prediktor se nazývá *spojitý*, jestliže může nabývat libovolných reálných hodnot.
2) Prediktor se nazývá *kategoriální*, jestliže může nabývat pouze hodnot z konečné množiny; prvky této množiny obvykle značíme přirozenými čísly 1, 2, ... . Dále rozlišujeme dva typy kategoriálních prediktorů.
   a) *Nominální* prediktor je kategoriální prediktor bez uspořádání kategorií.
   b) *Ordinální* prediktor je kategoriální prediktor s přirozeným uspořádáním kategorií.

Štěpení podle spojitého prediktoru $X_1$, který nabývá v uzlu $t$ hodnot z intervalu $(a, b)$, může vypadat takto:

$$sons(t) = \{s_1, \ldots, s_K\}, \quad \mathcal{A}(s_k) = \mathcal{A}(t) \cap \{x;\ h_{k-1} < x_1 \leqq h_k\},$$

kde $a = h_0 < h_1 < \cdots < h_K = b$ je nějaké dělení intervalu $(a, b)$; podle nominálního prediktoru $X_2$ takto:

$$\mathcal{A}(s_k) = \mathcal{A}(t) \cap \{x;\ x_2 \in A_k\},$$

kde $A_k$, $k = 1, \ldots, K$, je disjunktní rozklad množiny kategorií $\mathcal{C}_2(t)$, kterých může nabývat $X_2$ v uzlu $t$.

K dispozici je několik různých funkcí nečistoty. Budeme používat Giniho index

$$i(t) = \sum_{i \neq j} p(i|t)p(j|t).$$

Praxe ukazuje, že výsledný strom, především jeho chyba klasifikace, málo závisí na konkrétní volbě $i(t)$. Proto je pro kvalitu klasifikačního stromu kritické rozhodování, kdy uzel prohlásit za koncový a kdy ve štěpení pokračovat. Příliš velký strom bude pravděpodobně klasifikovat špatně nezávislou množinu dat, příliš malý zase nevyužije všechnu dostupnou informaci v $\mathcal{L}$.

**ad 2)**

K rozhodnutí o tom, kdy uzel prohlásit koncovým, použijeme metodu prořezávání. Spočívá v tom, že nejprve zkonstruujeme rozsáhlý strom $T_{max}$ štěpením uzlů, dokud koncové uzly neobsahují data pouze jedné třídy, nebo pouze málo dat. Pomocí vhodného odhadu $\widehat{R}(T)$ vybereme ten prořezaný podstrom $T_{opt} \preccurlyeq T_{max}$, který tento odhad minimalizuje.

### 3.2 Iterační algoritmus

Algoritmus iteračního růstu a prořezávání binárního stromu byl navržen v [5]. Tento algoritmus rozdělí data do dvou podsouborů a poté konstruuje binární strom na základě jednoho podsouboru a prořezává jej pomocí druhého, přičemž role obou podsouborů se střídají. Algoritmus upravíme pro konstrukci obecných stromů. Horní indexy u symbolů označují, ke kterému podsouboru se vztahují.

1) Náhodně rozdělíme učební soubor $\mathcal{L}$ na dva podsoubory $\mathcal{L}^{(1)}$ a $\mathcal{L}^{(2)}$ tak, že mají přibližně stejné počty dat z každé třídy vysvětlované proměnné $Y$. Horní indexy u již zavedených symbolů budeme používat k označení, ke kterému podsouboru se vztahují.
2) Použijeme $\mathcal{L}^{(1)}$ k vytvoření stromu $T_0$ postupným rozkládáním uzlů, až všechny koncové uzly $t \in \tilde{T}_0$ mají $N_j^{(1)}(t) = N^{(1)}(t)$ pro nějaké $j$, nebo $N^{(1)}(t) \leqq N_{min}$, nebo všechny prediktory všech dat mají stejné hodnoty, přestože data nepatří do jedné třídy. Dále přiřadíme každému uzlu tu třídu $j(t)$, která splňuje vztah $N_{j(t)}^{(1)}(t) = \max_{j \in \mathcal{C}} N_j^{(1)}(t)$.
3) Vybereme nejmenší prořezaný podstrom $T_0^* \preccurlyeq T_0$ takový, že

$$R^{(2)}(T_0^*) = \min_{T' \preccurlyeq T_0} R^{(2)}(T').$$

Položíme $k = 1$.

4) Položíme $\alpha = 1$ a $\beta = 2$, je-li $k$ sudé, resp. $\alpha = 2$ a $\beta = 1$, je-li $k$ liché.
5) Pomocí $\mathcal{L}^{(\alpha)}$ vytvoříme strom $T_k$ rozkladem koncových uzlů stromu $T_{k-1}^*$, až všechny koncové uzly $t \in \tilde{T}_k$ mají $N_j^{(\alpha)}(t) = N^{(\alpha)}(t)$ pro nějaké $j$, nebo $N^{(\alpha)}(t) \leqq N_{min}$, nebo hodnoty prediktorů stejné pro všechna data v $t$. Každému uzlu $t \in T_k - T_{k-1}^*$ přiřadíme tu třídu $j(t)$, která splňuje vztah $N_{j(t)}^{(\alpha)}(t) = \max_{j \in \mathcal{C}} N_j^{(\alpha)}(t)$.
6) Vybereme nejmenší prořezaný podstrom $T_k^* \preccurlyeq T_k$ takový, že

$$R^{(\beta)}(T_k^*) = \min_{T' \preccurlyeq T_k} R^{(\beta)}(T').$$

7) Jestliže $|T_k^*| = |T_{k-1}^*|$, pak položíme $T^* = T_k^*$ a proceduru ukončíme. Jinak položíme $k = k + 1$ a pokračujeme krokem 4.
8) Chybu klasifikace stromu $T^*$ odhadneme jako

$$\hat{R}(T^*) = \sum_{t \in U^{(1)}} R^{(2)}(t) + \sum_{t \in U^{(2)}} R^{(1)}(t),$$

kde $U^{(\beta)} = \{t \in \tilde{T}^*;$ uzlu $t$ byla přiřazena třída na základě $\mathcal{L}^{(\beta)}\}$, $\beta = 1, 2$.

Nechť $s(k,t)$, $\mathcal{A}(k,t)$ a $j(k,t)$ jsou štěpení, oblast a třída přiřazená uzlu $t \in T_k$ v $k$-tém iteračním kroku. Budeme předpokládat, že pokud $t \in T_k - \tilde{T}_k$, $t' \in T_l - \tilde{T}_l$, $\mathcal{A}(k,t) = \mathcal{A}(l,t')$ a $j(k,t) = j(l,t')$ a $t$ a $t'$ jsou štěpeny na základě stejného podsouboru, pak kdykoliv je to možné, volíme $s(k,t) = s(l,t')$, $j(k,s) = j(l,s')$, kde $s$ a $s'$ jsou odpovídající si děti uzlů $t$ a $t'$. Dále volíme $j(k,s) = j(k,t)$ pro $s \in sons(t)$, kdykoliv je to možné.

**Věta 3.2.1.**

$$T_{k-1}^* \preccurlyeq T_k^* \quad \forall k = 1, 2, \ldots.$$

*Tedy* $\mathcal{A}(k-1,t) = \mathcal{A}(k,t)$ a $j(k-1,t) = j(k,t)$ *pro všechna* $t \in T_{k-1}^*$ a $k = 1, 2, \ldots$.

**Věta 3.2.2.**

(a) *Nechť* $K(t) = \inf\{k \geqq 1;\ t \in \tilde{T}_{k-1}^* \cap \tilde{T}_k^*\}$ *pro* $t = 1, 2, \ldots$. *Je-li* $K(t) < \infty$, *pak* $t \in \tilde{T}_k^*$ $\forall k \geqq K(t)$. *Tedy* $\mathcal{A}(k,t) = \mathcal{A}(K(t),t)$ a $j(k,t) = j(K(t),t)$ $\forall k \geqq K(t)$.
(b) *Nechť* $K = \inf\{k \geqq 1;\ |T_{k-1}^*| = |T_k^*|\}$. *Pak* $K < \infty$ a $T_k^* = T_K^*$ $\forall k \geqq K$. *Tedy také* $\mathcal{A}(k,t) = \mathcal{A}(K,t)$ a $j(k,t) = j(K,t)$ $\forall k \geqq K$.

*Důkaz.* Důkazy těchto vět pro striktně binární stromy lze nalézt v [5]. Jejich zobecnění pro nebinární stromy lze nalézt v [7].

**Poznámky.**

- Věta 3.2.1 říká, že posloupnost optimálně prořezaných stromů je vnořená a neklesající co do velikosti.
- Část (a) věty 3.2.2 říká, že jestliže uzel je koncový ve dvou po sobě následujících optimálně prořezaných stromech, je pak koncový již ve všech následujících.
- Část (b) věty 3.2.2 nás ujišťuje, že algoritmus je konečný a skončí právě tehdy, jakmile je počet uzlů ve dvou po sobě následujících stromech stejný.

### 3.3 Prořezávání stromu

Nyní popišme algoritmus prořezávání stromu $T$. Označme $c = |T|$ a nechť uzly v $T$ mají čísla $t_1 < t_2 < \cdots < t_c$. Algoritmus spočívá v tom, že porovnává chybu klasifikace spočtenou na základě $\mathcal{L}^{(\beta)}$ v uzlu $t$ a ve větvi $T_t$. Pokud je $R^{(\beta)}(t) \leqq R^{(\beta)}(T_t)$, je větev $T_t$ nahrazena samotným uzlem $t$. Definujeme proto $S(t) = R(T_t)$ (vynecháváme již index $\beta$). Protože čísla jsou přiřazována uzlům tím způsobem, že čísla následníků jsou větší než čísla předchůdců, stačí nám k prořezání jediný průchod stromem ze zdola nahoru.

Pro $t = t_c, \ldots, t_1$ dělej

{Jestliže $t \in \tilde{T}$ pak

$\{S(t) := R(t)\}$

Jestliže $t \in T - \tilde{T}$ pak

$\{S(t) := \sum_{s \in sons(t)} S(s);$

Jestliže $R(t) \leqq S(t)$ pak

$\{T := T - \bigcup_{s \in sons(t)} T_s;$

$sons(t) := \{\};$

$S(t) := R(t)$

$\}$

$\}$

$\}$

Tento algoritmus tedy prochází stromem od listů ke kořenu. Po každé, kdy je odříznuta nějaká větev $T_t$ a nahrazena uzlem $t$, jsou modifikovány hodnoty $sons(t)$ a $S(t)$.

## 4. NEBINÁRNÍ ŠTĚPENÍ

V této kapitole se zaměříme na konstrikci a volbu štěpení. Omezíme se na kategoriální prediktory.

### 4.1 Konstrukce štěpení

K hledání rozkladu máme v uzlu $t$ k dispozici $N^{(\alpha)}(t)$ dat z $\mathcal{L}^{(\alpha)}$. V uzlu $t$ může $m$-tý prediktor nabývat hodnot z množiny $C_m(t)$ a nechť bez újmy na obecnosti $C_m(t) = \{1, 2, \ldots, J_m\}$. Položíme $n_{lj}^{(m)}(t) = |\{x_n \in \mathcal{A}(t) \cap \mathcal{L}^{(\alpha)};\ (x_n)_m = l,\ y_n = j\}| \equiv$ počet dat z $\alpha$-tého podsouboru v uzlu $t$, jejichž $m$-tý prediktor má hodnotu $l \in C_m(t)$ a jenž patří do $j$-té třídy, $j \in \mathcal{C}$. Označme

$$n_{l.}^{(m)}(t) = \sum_{j=1}^{J} n_{lj}^{(m)}(t),$$

$$n_{.j}^{(m)}(t) = \sum_{l \in C_m(t)} n_{lj}^{(m)}(t) = N_j^{(\alpha)}(t),$$

$$n_{..}^{(m)}(t) = \sum_{j=1}^{J} \sum_{l \in C_m(t)} n_{lj}^{(m)}(t) = \sum_{j=1}^{J} N_j^{(\alpha)}(t) = N^{(\alpha)}(t).$$

Tímto způsobem obdržíme pro každý prediktor kontingenční tabulku:

| $X_m$ | $Y$ 1 | 2 | ... | $J$ | $\sum$ |
|---|---|---|---|---|---|
| 1 | $n_{11}^{(m)}(t)$ | $n_{12}^{(m)}(t)$ | ... | $n_{1J}^{(m)}(t)$ | $n_{1.}^{(m)}(t)$ |
| 2 | $n_{21}^{(m)}(t)$ | $n_{22}^{(m)}(t)$ | ... | $n_{2J}^{(m)}(t)$ | $n_{2.}^{(m)}(t)$ |
| $\vdots$ | ... | ... | ... | ... | $\vdots$ |
| $J_m$ | $n_{J_m1}^{(m)}(t)$ | $n_{J_m2}^{(m)}(t)$ | ... | $n_{J_mJ}^{(m)}(t)$ | $n_{J_m.}^{(m)}(t)$ |
| $\sum$ | $n_{.1}^{(m)}(t)$ | $n_{.2}^{(m)}(t)$ | ... | $n_{.J}^{(m)}(t)$ | $n_{..}^{(m)}(t)$ |

Tabulka 1. Kontingenční tabulka $X_m \times Y$.

Je zřejmé, že celkový počet dat a sloupcové marginální součty budou pro všechny prediktory stejné, ale lišit se mohou počty řádků a řádkové marginální součty.

V případě ordinálního prediktoru jsou povoleny pouze takové rozklady, kdy jsou sdružovány sousední kategorie. Počet možných rozkladů $w_{ord}(L,K)$ $L$ kategorií na $K$ neprázdných skupin je dán vzorcem

$$w_{ord}(L,K) = \binom{L-1}{K-1},$$

což dává například:

| K | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| L = 8 | 7 | 21 | 35 | 35 |
| L = 20 | 19 | 171 | 969 | 3876 |

Tabulka 2. $w_{ord}(L,K)$ pro vybraná $L$ a $K$.

V případě nominálního prediktoru je počet $w_{nom}$ možných rozkladů $L$ kategorií do $K$ neprázdných skupin dán vzorcem

$$w_{nom} = \frac{1}{K!}\sum_{i=0}^{K}\binom{K}{i}(-1)^i(K-i)^L,$$

což dává mnohem větší čísla, například:

| K | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| L = 8 | 127 | 966 | 1701 | 1050 |
| L = 20 ($\times 10^6$) | 0.524 | 580 | 45232 | 749206 |

Tabulka 3. $w_{nom}(L,K)$ pro vybraná $L$ a $K$.

Pro ordinální prediktor a malá $L$ je tedy možné zkoušet všechny možnosti a vybrat ten rozklad $A_1, A_2, \ldots, A_K$, který minimalizuje nečistotu

$$\sum_{k=1}^{K} \left[ \frac{\sum_{l \in A_k} n_{l.}^{(m)}(t)}{n_{..}^{(m)}(t)} \left( 1 - \sum_{j=1}^{J} \left( \frac{\sum_{l \in A_k} n_{lj}^{(m)}(t)}{\sum_{l \in A_k} n_{l.}^{(m)(t)}} \right)^2 \right) \right].$$

Pro nominální prediktor je časově nemožné zkoušet všechny možnosti k nalezení nejlepší, proto se uchýlíme k hledání minima lokálního. Použijeme algoritmus ze shlukové analýzy popsaný v [2].

Nechť $n_{l.}^{(m)} \neq 0$ pro všechna $l \in C_m(t)$. Chceme najít disjunktní rozklad množiny $C_m(t)$ hodnot $m$-tého prediktoru na množiny $A_1, A_2, \ldots, A_K$ a jim odpovídající uzly $t_1, t_2, \ldots, t_K$, který minimalizuje průměrnou nečistotu

$$I(A_1, \ldots, A_K|t) = \sum_{k=1}^{K} p(t_k|t) i(t_k),$$

kde $p(t_k|t)$ označuje pravděpodobnost, že objekt v uzlu $t$ má hodnotu $m$-tého prediktoru v $A_k$. Tu odhadneme výrazem

$$p(t_k|t) = \frac{\sum_{l \in A_k} n_{l.}^{(m)}(t)}{n_{..}^{(m)}(t)}.$$

Analogicky označme

$$p(s_l|t) = \frac{n_{l.}^{(m)}(t)}{n_{..}^{(m)}(t)}$$

odhad pravděpodobnosti, že objekt v uzlu $t$ má hodnotu $m$-tého prediktoru rovnu $l$. To znamená, že $s_l$, $l \in C_m(t)$, odpovídá rozkladu, kdy přiřadíme syna každé hodnotě $l \in C_m(t)$.

Označme $\mu(\tau) = (p(1|\tau), p(2|\tau), \ldots, p(J|\tau))'$ vektor relativních četností jednotlivých tříd v uzlu $\tau$; $\mu$ budeme nazývat střed uzlu. Dále označme $d(\tau_1, \tau_2) = \delta(\mu(\tau_1), \mu(\tau_2)) = \sum_{j=1}^{J} (\mu_j(\tau_1) - \mu_j(\tau_2))^2$ vzdálenost středů uzlů $\tau_1$ a $\tau_2$.

**Věta 4.1.1.** *Nutná podmínka, aby rozklad $A_1, \ldots, A_K$ množiny $C_m(t)$ měl minimální průměrnou nečistotu $I(A_1, \ldots, A_K|t) = \sum_{k=1}^{K} p(t_k|t) i(t_k)$ je taková, že $l \in A_k$ jen když $k = \arg\min_{l \in C_m(t)} d(s_l, t_k)$ nebo když $p(s_l|t) = 0$, kde $t_k$ je určeno podle $A_k$.*

*Důkaz.* Tato věta je dokázána v [2] pro $K = 2$ a důkaz pro libovolné $K \geq 2$ lze nalézt v [7].

Nyní můžeme navrhnout algoritmus, který v každém kroku snižuje průměrnou nečistotu.

1) Zvolíme výchozí seskupení $A_1, A_2, \ldots, A_K$ kategorií z $C_m(t)$ do $K$ skupin.

2) Spočítáme středy těchto skupin jako vektory relativních četností jednotlivých tříd, tj.

$$\mu(k) = (\mu_1(k), \mu_2(k), \ldots, \mu_J(k))',$$

kde

$$\mu_j(k) = \frac{\sum_{l \in A_k} n_{lj}^{(m)}(t)}{\sum_{l \in A_k} n_{l.}^{(m)}(t)}, \qquad k = 1, 2, \ldots, K, \quad j = 1, 2, \ldots J.$$

3) Iteračně opakujeme následující kroky:
   a) Opravíme rozklad tak, že každý řádek relativních četností v kontingenční tabulce přiřadíme do té skupiny, k jejímuž středu je nejblíže, tj.

$$A_k = \left\{ l \,\middle|\, k = \arg\min_{\kappa = 1, \ldots, K} \sum_{j=1}^{J} \left( \frac{n_{lj}^{(m)}(t)}{n_{l.}^{(m)}(t)} - \mu_j(\kappa) \right)^2 \right\}$$

   pro $k = 1, 2, \ldots, K$, přičemž v případě stejné vzdálenosti od více středů volíme přiřazení libovolně.
   b) Spočítáme středy nově vytvořených skupin.
4) Iteraci ukončíme, jakmile nebude zaznamenán žádný pokles nečistoty rozkladu. Tu počítáme pomocí Giniho indexu jako

$$\sum_{k=1}^{K} \left[ \frac{\sum_{l \in A_k} n_{l.}^{(m)}(t)}{n_{..}^{(m)}(t)} \left( 1 - \sum_{j=1}^{J} (\mu_j(k))^2 \right) \right].$$

### 4.2 Volba štěpení

Teď jsme schopni konstruovat štěpení s libovolným počtem skupin pro oba typy prediktorů. Můžeme vyzkoušet všechny binární štěpení a vybrat $s_2$ s minimální nečistotou, řekněme $I_2$. Podobně můžeme vybrat nejčistší 3-, 4-, ... , $MaxK$-cestná štěpení $s_3, s_4, \ldots, s_{MaxK}$ s nečistotami $I_3, I_4, \ldots, I_{MaxK}$. Zřejmě $I_2 \geqq I_3 \geqq \cdots \geqq I_{MaxK}$.

Měli bychom tedy použít $s_{MaxK}$ ke štěpení $t$? Můžeme očekávat velký pokles nečistoty na začátku posloupnosti $I_2, I_3, \ldots, I_{MaxK}$ a jen malé změny na jejím konci. Navíc, jestliže se rozhodneme pro příliš jemné štěpení, obdržíme uzly s mnohem méně daty, než bylo v $t$, a bude poměrně obtížné pokračovat v jejich štěpení. Na druhou stranu třícestná nebo čtyřcestná štěpení mohou někdy lépe vystihnout strukturu dat než binární štěpení.

Zásadní otázka zní: *Kolik je správný počet skupin ve štěpení?* Budeme potřebovat nějakou metodu pro porovnávání štěpení s různým počtem skupin. Nechť $MaxK$ označuje maximální uvažovaný počet skupin ve štěpení. Nechť $s_k$ je štěpení s $k$ skupinami. Podle $s_k$ vytvoříme syny $t$ a budeme je dále štěpit, až dostaneme strom s kořenem $t$ a $MaxK$ listy. Jako nejlepší štěpení vybereme to, které vedlo k nejčistšímu stromu. Ještě musíme uvážit, jaká štěpení budeme povolovat jako počáteční a pomocí kterých štěpení je pak budeme doplňovat.

Základní možnosti pro počáteční štěpení:

1) Zkoušet všechna štěpení.
2) Pro každý prediktor $X_m$, $m = 1, \ldots, M$, a pro $K = 2, 3, \ldots, MaxK$ vybírat štěpení podle $X_m$ s $K$ skupinami a minimální nečistotou.
3) Pro $K = 2, 3, \ldots, MaxK$ vybírat štěpení s $K$ skupinami a minimální nečistotou.

Základní možnosti pro doplňování štěpení:

1) Všechny možné rozklady.
2) Postupně štěpit dočasné listy, až počet koncových uzlů dosáhne $MaxK$. Pro štěpení je možno používat:
   a) pouze binární štěpení;
   b) jak binární, tak nebinární štěpení.

Podívejme se nyní na posloupnost nejčistších štěpení $s_2, \ldots, s_{MaxK}$ s $2, \ldots, MaxK$ skupinami podle $m$-tého prediktoru. Je docela možné, že $s_k$ je pouhým zjemněním $s_l$, $l < k$. Jsou-li například kategorie $m$-tého prediktoru rozděleny štěpením $s_2$ na skupiny $\{1,2,3\}$ a $\{4,5,6,7,8\}$, štěpením $s_3$ na skupiny $\{1,2,3\}$, $\{4,5\}$ a $\{6,7,8\}$ a štěpením $s'_3$ na skupiny $\{1,2\}$, $\{3,4,5\}$ a $\{6,7,8\}$, je $s_3$ zjemněním $s_2$, to však neplatí o $s'_3$. V případě, že $s_k$ je zjemněním některého ze štěpení $s_2, \ldots, s_{k-1}$, řekněme $s_l$, jen stěží můžeme očekávat, že by $s_k$ bylo doplněno lépe než $s_l$. Proto budeme šetřit čas a taková štěpení vyloučíme z procesu doplňování.

Když uvažujeme nebinární štěpení v kontextu Gelfandova algoritmu, narazíme na nový problém. Představme si, že uzel $t$ je postupně štěpen binárními štěpeními. Při prořezávání části stromu $T_t$ je vybrán podstrom $T'_t \preccurlyeq T_t$. Počet koncových listů v $T'_t$ může být 1 až $|\tilde{T}_t|$. Předpokládejme, že všechna štěpení použitá pro konstrukci $T_t$ jsou založena na hodnotách jediného prediktoru. Pak lze tato štěpení alternativně vyjádřit jako jedno vícecestné štěpení. Avšak nyní jsme již ztratili rozmanitost volby při prořezávání. Můžeme pouze toto vícecestné štěpení ve stromě ponechat nebo ho celé odstranit. Proto je třeba vícecestná štěpení zařazovat do stromu velice uvážlivě a opatrně. Pokud máme posloupnost navržených štěpení $s_2, \ldots, s_{MaxK}$ podle $m$-tého prediktoru, rozdělíme data z $t$ podle každého štěpení a v každé skupině vybereme nejčetnější kategorii $Y$. Poté spočítáme počty $r_2, \ldots, r_{MaxK}$ špatně klasifikovaných dat z $\mathcal{L}^{(\beta)}$ štěpeními $s_2, \ldots, s_{MaxK}$. Nalezneme největší $K_0 \in \{2, \ldots, MaxK\}$ takové, že $r_{K_0} = \min_{k=2,\ldots,MaxK} r_k$ a vyloučíme štěpení $s_{K_0+1}, \ldots, s_{MaxK}$ z dalších úvah.

Pro realizaci popsané metody jsme zvolili možnost 2) pro počáteční štěpení a možnost 2a) pro doplňování. Uzel s takovými daty, že všechny prediktory mají stejné hodnoty, ale data nepatří do stejné třídy vysvětlované proměnné, budeme pro stručnost nazývat *degenerovaným*. Nyní popíšeme algoritmus doplňování uzlu $s_K$. Algoritmus konstruuje strom $T_t$ s kořenem $t$ a, pokud je to možné, s $MaxK$ listy, spočítá jeho nečistotu a poté následníky $t$ zruší.

```
Rozštěp t na syny t_1,...,t_K podle s_K;
KeŠtěpení := {t_1,...,t_K};
Z KeŠtěpení vyluč čisté a degenerované uzly;
UvažovanáŠtěpení := {};
Pro každý uzel v KeŠtěpení dělej
    {
     Najdi nejčistší binární štěpení;
     Přidej jej do UvažovanáŠtěpení;
     Spočti jeho nečistotu
    }
i := K + 1;
Dokud (UvažovanáŠtěpení ≠ ∅) and (i ≦ MaxK) dělej
    {
```

```
      Vyber z UvažovanáŠtěpení štěpení s
          s největším poklesem nečistoty;
      Realizuj s a dva nové uzly označ τ1, τ2;
      Pro τ := τ1, τ2 dělej
           Jestliže τ není degenerované nebo čisté pak
              {
               Najdi nejlepší binární štěpení uzlu τ;
               Přidej jej do UvažovanáŠtěpení;
               Spočti jeho nečistotu
              }
      i := i + 1;
      Odeber s z UvažovanáŠtěpení
    }
  Spočti nečistotu zkonstruovaného stromu jako součet nečistot
     jeho listů s vahami rovnými počtu dat v každém z nich;
  Zruš všechny následníky t.
```

Zbývá nám popsat algoritmus výběru štěpení.

```
  NejmenšíNečistota := 1;
  Pro každý prediktor dělej
      {
       Pro K := 2 do MaxK dělej
            {
             Najdi nejčistší štěpení s_K s K skupinami;
             Spočti počet chybně klasifikovaných dat r_K v L^(β)
               při štěpení s_K.
            }
       Najdi největší K0 takové, že r_K ≧ r_K0 ∀ K < K0;
       Nečistota := 1;
       Pro K := 2 do K0 dělej
            {
             Doplň s_K a vypočtenou hodnotu
                 nečistoty vlož do NováNečistota;
             Jestliže NováNečistota ≦ Nečistota pak
                  {
                   Nečistota := NováNečistota;
                   Štěpení := s_K
                  }
            }
       Jestliže Nečistota < NejmenšíNečistota pak
            {
             NejmenšíNečistota := Nečistota;
             VybranéŠtěpení := Štěpení
            }
      }
```

Algoritmus nejprve zkonstruuje nejčistší 2-, ..., $MaxK$-cestná štěpení podle kaž-

dého prediktoru, doplní je pomocí výše popsaného algoritmu a vybere to štěpení, které vedlo k nejčistšímu stromu. Vybrané štěpení je uloženo v proměnné *VybranéŠtěpení* a odpovídající nečistota v *NejmenšíNečistota*.

## 5. PŘÍKLAD

Proces volby štěpení si ilustrujeme na malém příkladě. Uzel $t$ bylo možné rozdělit buď pomocí prediktoru $X_1$ s devíti kategoriemi nebo pomocí prediktoru $X_2$ se čtyřmi kategoriemi. Oba prediktory byly ordinální. Kontingenční tabulka měla tento tvar:

| $X_2$ | $Y$ | $X_1$ 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 2 | 0 | 0 | 2 | 0 | 1 | 0 | 1 | 0 | 0 |
| | 3 | 0 | 0 | 0 | 1 | 2 | 2 | 3 | 3 | 5 |
| | 4 | 0 | 0 | 0 | 0 | 0 | 1 | 2 | 3 | 6 |
| 2 | 1 | 1 | 1 | 2 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 2 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 2 | 1 |
| | 3 | 0 | 0 | 0 | 2 | 2 | 3 | 3 | 3 | 3 |
| | 4 | 0 | 0 | 0 | 1 | 2 | 2 | 2 | 3 | 7 |
| 3 | 1 | 0 | 1 | 2 | 1 | 0 | 0 | 1 | 0 | 0 |
| | 2 | 0 | 0 | 2 | 5 | 4 | 4 | 1 | 1 | 0 |
| | 3 | 0 | 0 | 0 | 0 | 1 | 3 | 2 | 3 | 2 |
| | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 2 | 2 |
| 4 | 1 | 0 | 0 | 1 | 2 | 1 | 1 | 0 | 0 | 0 |
| | 2 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| | 3 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Tabulka 4. Kontingenční tabulka $X_1 \times X_2 \times Y$.

Nyní můžeme spočítat kontingenční tabulky $X_1 \times Y$ and $X_2 \times Y$:

| $X_2$ | $Y$ 1 | 2 | 3 | 4 | $\sum$ |
|---|---|---|---|---|---|
| 1 | 2 | 4 | 16 | 12 | 34 |
| 2 | 4 | 6 | 16 | 17 | 43 |
| 3 | 5 | 17 | 11 | 5 | 38 |
| 4 | 5 | 1 | 1 | 0 | 7 |
| $\sum$ | 16 | 28 | 44 | 34 | 122 |

Tabulka 5. Kontingenční tabulka $X_2 \times Y$.

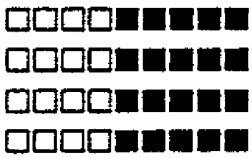
| $X_1$ | $Y$ 1 | 2 | 3 | 4 | $\sum$ |
|---|---|---|---|---|---|
| 1 | 2 | 0 | 0 | 0 | 2 |
| 2 | 3 | 0 | 0 | 0 | 3 |
| 3 | 5 | 4 | 1 | 0 | 10 |
| 4 | 3 | 7 | 3 | 1 | 14 |
| 5 | 1 | 5 | 5 | 2 | 13 |
| 6 | 1 | 5 | 8 | 3 | 17 |
| 7 | 1 | 3 | 8 | 5 | 17 |
| 8 | 0 | 3 | 9 | 8 | 20 |
| 9 | 0 | 1 | 10 | 15 | 26 |
| $\sum$ | 16 | 28 | 44 | 34 | 122 |

Tabulka 6. Kontingenční tabulka $X_1 \times Y$.

Podívejme se na případy $MaxK = 2$ a $MaxK = 3$.

1) $MaxK = 2$

V tomto případě musíme porovnat nejčistší binární štěpení podle $X_1$ a nejčistší binární štěpení podle $X_2$. Rozdílné skupiny jsou vyjádřeny rozdílnými symboly □, ■ a ⊠. Symbol v $i$-tém řádku a $j$-tém sloupci vyjadřuje, ke které skupině patří $i$-tá kategorie $X_2$ a $j$-tá kategorie $X_1$.

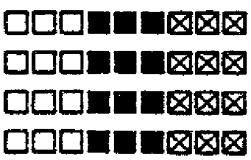


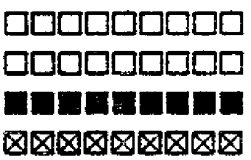
nečistoty: 1) 0.649963 2) 0.678977

Je zřejmé, že bude vybráno štěpení podle $X_1$.

2) $MaxK = 3$

Nyní budeme porovnávat čtyři rozklady: nejčistší třícestné štěpení podle $X_1$, nejčistší třícestné štěpení podle $X_2$ a nejlepší doplnění binárních štěpení z případu $MaxK = 2$. Na obrázcích jsou znázorněny tyto situace: nejčistší třícestné štěpení podle $X_1$, nejčistší třícestné štěpení podle $X_2$, nejlepší doplnění nejčistšího binárního štěpení podle $X_1$ štěpením podle $X_2$, nejlepší doplnění nejčistšího binárního štěpení podle $X_2$ štěpením podle $X_1$, nejlepší doplnění nejčistšího binárního štěpení podle $X_1$ štěpením podle $X_1$, kterážto nemůže být lepší než nejlepší třícestné štěpení podle $X_1$, ale je lepší než nejlepší doplnění štěpením podle $X_2$. Poslední obrázek ukazuje skutečně nejčistší rozklad na tři skupiny, ale tento rozklad nebyl uvažován, a pravděpodobně by mohl být nalezen pouze prohledáváním všech možností.





nečistoty: 3) 0.619819 4) 0.656152

nečistoty: 5) 0.626758 6) 0.621850

nečistoty: 7) 0.626396 8) 0.612436

Jestliže $MaxK = 3$, algoritmus vybere třícestné štěpení podle $X_1$.

Při $MaxK = 2$ je uzel $t$ nejprve rozštěpen na uzly $t_1$ a $t_2$. Data s $X_1 \in \{1,2,3,4\}$ jsou přiřazena do $t_1$ a data s $X_1 \in \{5,6,7,8,9\}$ jsou přiřazena do $t_2$. V dalším kroku by byl uzel $t_2$ rozštěpen na uzly $t_3$ a $t_4$. Data s $X_1 \in \{5,6,7\}$ by byla přiřazena do $t_3$ a data s $X_1 \in \{8,9\}$ by byla přiřazena do $t_4$. Tato situace odpovídá obrázku 7 s nečistotou 0.626396. Je zřejmé, že v obecném případě větší $MaxK$ umožňuje vybrat $MaxK$-cestné štěpení nebo upřednostnit jiné $K$-cestné štěpení, $K < MaxK$, než při běhu algoritmu s menším $MaxK$.

## 6. PROGRAM

K popsanému algoritmu byl vytvořen počítačový program. Jedná se o cca 1800 řádků v programovacím jazyce *Mathematica*. Tento program zahrnuje procedury pro konstrukci klasifikačního stromu, grafický výstup výsledného stromu, kategorizaci spojitých proměnných, výpočet důležitosti prediktorů, klasifikaci nezávislého souboru dat, odhad chyby klasifikace, atd.

## 7. SHRNUTÍ

V této práci byla popsána metoda rozšíření Gelfandova algoritmu o nebinární štěpení. Jaké jsou výhody tohoto přístupu? Omezili jsme se na kategoriální prediktory. Pokud by proměnná byla spojitá, bylo by nutné ji nejprve kategorizovat. To není příliš omezující, neboť cílem konstrukce klasifikačního stromu je právě seskupování dat.

Rozšíření množiny uvažovaných štěpení a jejich doplňování způsobují, že algoritmus pracuje výrazně pomaleji při větším $MaxK$. Avšak čas nutný pro konstrukci stromu není obvykle tak důležitý a na druhou stranu můžeme těžit z trochu přesnějších výsledných stromů během klasifikace nových pozorování. Samozřejmě nikde není zaručeno, že s větším $MaxK$ obdržíme lepší strom. Jestliže se nějaké štěpení zdá lepší než jiné po doplnění na větev s $K$ listy, může být situace opačná ve výsledném stromu. Skutečná chyba klasifikace může být dobře odhadnuta na základě rozsáhlého nezávislého souboru dat. Větší pokles nečistoty v jedné fázi konstrukce stromu zdaleka neznamená menší skutečnou chybu klasifikace.

Lze se domnívat, že stromy konstruované popsaným algoritmem by měly být v průměru přesnější. Tuto domněnku lze pravděpodobně ověřit pouze empiricky na základě analýzy mnoha souborů dat porovnáním závislosti výsledků na zvoleném $MaxK$.

## LITERATURA

1. Breiman, L., Friedman, J. H., Olshen, R. A., Stone, C. J., *Classification and Regression Trees,*

The Wadsworth Statistics/Probability Series, Belmont, CA : Wadsworth, 1984.

2. Chou, P. A., *Optimal Partitioning for Classification and Regression Trees*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 13, No. 4 (1991), 340–353.
3. Ciampi, A., *Constructing Prediction Trees from Data, The RECPAM Approach*, Antoch, J. (editor), Computational Aspects of Model Choice, (1993), Physica – Verlag, Heidelberg.
4. Duran, B. S., Odell, P. L., *Cluster Analysis*, Springer-Verlag Berlin, 1974.
5. Gelfand, S. B., Ravishankar, C. S., Delp, E. J., *An iterative growing and pruning agorithm for classification tree design*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 13, No. 2 (1991), 163–174.
6. Hawkins, D. M., *FIRM (Formal Inference-based Recursive Modeling)* (1990), Technical Report Number 546, University of Minnesota, School of Statistics.
7. Keprta, S., *Klasifikátory konstruované pomocí rekurzivních postupů*, Diplomová práce, 1993.
8. Mingers, J., *Emperical comparison of selection measures for decision tree induction*, Machine Learning vol. 3 (1989), 319–342.
9. Quinlan, J. R., *Induction of decision trees*, Machine Learning vol. 1, no. 1 (1986), 81–106.
10. Safavian, S. R., Landgrebe, D., *A Survey of Decision Tree Classifier*, IEEE Transactions on Systems, Man, and Cybernetics, vol. 21, No. 3 (1991).
11. Wolfram, S., *Mathematica: A System for Doing Mathematics by Computer*, Addison – Wesley Publishing Co, Redwood City, California, U.S.A., 1991.

MFF UK, KPMS, Sokolovská 83, 186 00 PRAHA 8
*E-mail address*: keprta@karlin.mff.cuni.cz