Tvorba klasifikačných stromov pri procese data miningu

BAKALÁRSKA PRÁCA

Ivan Sutóris

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY
KATEDRA APLIKOVANEJ MATEMATIKY A ŠTATISTIKY

Študijný program: 9.1.9 Aplikovaná matematika
Študijný odbor: Ekonomická a finančná matematika

Vedúci záverečnej práce:
RNDr. Igor Odrobina, CSc.

BRATISLAVA 2007

Čestne prehlasujem, že som bakalársku prácu vypracoval samostatne, len s pomocou nadobudnutých vedomostí, konzultácií a uvedenej literatúry.

Bratislava, jún 2007 Ivan Sutóris

Chcem sa poďakovať vedúcemu práce RNDr. Igorovi Odrobinovi Csc. za jeho čas, pomoc a pripomienky, ktorými mi pomohol pri vypracovaní práce. Zároveň ďakujem mojej rodine a priateľom za ich podporu.

Abstrakt

Pojmami data mining, alebo hľadanie znalostí v databázach sa označuje v súčasnosti populárna problematika na pomedzí štatistiky a informatiky, zaoberajúca sa nachádzaním vzťahov a vzorcov vo veľkých objemoch dát. Jednou z metód používaných pri data miningu sú klasifikačné (nazývané aj rozhodovacie) stromy, ktorými sa zaoberá táto práca. Ťažiskom práce je všeobecný opis základných algoritmov na tvorbu klasifikačných stromov a podrobnejší popis konkrétneho algoritmu CART (classification and regression trees). Súčasťou práce je aj niekoľko príkladov ilustrujúcich uvádzanú teóriu, ktoré sú realizované v štatistickom prostredí R.

Kľúčové slová:

data mining, objavovanie znalostí v databázach, klasifikačné stromy, rozhodovacie stromy, CART

Abstract

Data mining, or knowledge-discovery in databases, is recently popular field with links to disciplines such as statistics and computer science, which is concerned with finding patterns and relations in large volumes of data. One of methods used in data mining are classification (sometimes called decision) trees, which are the main subject of this work. Focus is on general description of basic algorithms for building classification trees and more detailed description of particular algorithm CART (classification and regression trees). Several examples illustrating the theory, implemented in statistical enviroment R, are included.

Keywords:

data mining, knowledge discovery in databases, classification trees, decision trees, CART

Obsah

Zoznam obrázkov

Úvod

Pojmom data mining, inak nazývaným aj „objavovanie znalostí v databázach“, (knowledge discovery in databases), sa označuje súbor techník a algoritmov, používaných na (polo-) automatizované nachádzanie vzťahov a vzorcov vo veľkých objemoch dát. V dnešnej informatizovanej dobe majú nové poznatky a informácie často veľkú hodnotu, preto je táto problematika čoraz populárnejšia. Postupy data mininu sa používajú v mnohých oblastiach, zaujímavými aplikáciami sú okrem iných napríklad marketing (predpovedanie a hľadanie najviac prínosných zákazníkov), detekcia finančných podvodov (odhalenie nezvyklých transakcií), alebo spracovanie vedeckých dát (napr. astronomické zábery).

Jednou z metód používaných pri data miningu sú modely v podobe klasifikačných stromov, ktoré slúžia na klasifikáciu alebo predikciu vlastností objektov na základe iných ich vlastností. Klasifikáciou sa myslí zaradenie objektu podľa jeho vlastností do niektorej zo skupiny tried (teda premenná, ktorú predpovedáme, je diskrétna), predikciou zase predpovedanie numerickej vlastnosti objektu (predpovedaná premenná je spojitá)[1]. Pre vybudovanie modelu je (podobne ako pri iných data miningových technikách) potrebná množina objektov, pri ktorých už poznáme hodnotu hľadanej vlastnosti (trénovacia množina). Výsledkom je súbor pravidiel pre klasifikáciu/predikciu, ktoré sa dajú zobraziť v podobe stromu. Výhodou klasifikačných stromov je práve názornosť a zrozumiteľnosť výsledného modelu, vďaka čomu sa často používajú v praxi.

V našej práci sa snažíme podať základný opis algoritmov na budovanie klasifikačných stromov. V kapitole 1 je uvedený stručný súhrn rozličných metód používaných v data miningu. Kapitola 2 obsahuje všeobecný popis hlavných prvkov algoritmov pre klasifikačné stromy, zatiaľ čo v 3. kapitole sa venujeme podrobnejšie jednému konkrétnemu algoritmu s názvom CART. Náplňou kapitoly 4 je niekoľko praktických príkladov, na ktorých demonštrujeme funkčnosť algoritmu CART vo voľne šíriteľnom štatistickom prostredí R s použitím knižnice rpart.

1 Niekedy sa tieto pojmy chápu odlišne – klasifikáciou sa myslí skôr deskriptívny popis súčasných dát a predikciou predpovedanie pre budúce dáta, v tejto práci sa však budeme pridržať uvedenej terminológie.

1 Data Mining

Data mining je multidisciplinárnym procesom na rozhraní štatistiky a informatiky (hlavne v oblasti umelej inteligencie a strojového učenia). Ide pritom o pomerne mladú disciplínu, ktorá sa rozvíja najmä v posledných desaťročiach. Hlavnými motívmi tohto rozvoja sú rastúci výpočtový výkon súčasných počítačov, umožňujúci analýzu čoraz väčšieho počtu dát, a stúpajúce množstvo údajov uložených v databázach. Cieľom data miningu je objavovanie nových poznatkov, ktoré by mali byť štatisticky platné a potenciálne užitočné. Typickými úlohami data miningu sú klasifikácia, predikcia (viď úvod) alebo deskriptívny popis dát odhaľujúci nové vzťahy. Na vybudovanie modelov, ktoré nám umožnia klasifikáciu alebo predikciu, využívame databázu objektov, u ktorých už hľadanú triedu/vlastnosť poznáme, trénovaciu množinu.

Medzi používané metódy v data miningu (viď napr. publikáciu [4]) patria napríklad:

- Klasické štatistické techniky, ako lineárna regresia, používaná na predikciu numerickej premennej, alebo logistická regresia na predpovedanie kategorickej, binárnej premennej.
- Clustering, ktorý sa zaoberá rozdelením objektov do skupín, v rámci sú z hľadiska vlastností čo najbližšie k sebe. Na meranie vzájomnej vzdialenosti objektov podľa ich vlastností je potrebné definovať metriku (ak sú všetky vlastnosti vyjadrené číselnými premennými, môže ísť napr. o jednoduchú euklidovskú normu).
- Metóda k-najbližších susedov sa snaží predpovedať vlastnosť objektu na základe iných objektov čo najbližšie k nemu. Vzdialenosť medzi objektami sa definuje podobne ako pri clusteringu.
- Klasifikačné stromy, ktoré sú témou tejto práce.
- Neurónové siete sú matematickým modelom napodobňujúcim fungovanie mozgu. Skladajú sa zo siete uzlov, ktoré majú stanovenú váhu a sú navzájom prepojené jednoduchými aritmetickými operáciami. Dáta vstupujú do vstupných uzlov a pomocou týchto operácii sieť vyráta výsledok vo výstupných uzloch. Pri budovaní modelu na trénovacej množine sa predpovedaný výsledok porovná so skutočným a na základe toho sa zakaždým upravuje váha jednotlivých uzlov. Výsledkom môže byť model, ktorý má dobrú predikčnú schopnosť, nevýhodou sú však ťažkosti s jeho interpretáciu.
- Asociačné pravidlá extrahujú z databázy množstvo vzťahov medzi objektami a následne sa snažia identifikovať tie, ktoré sú najpresnejšie a zároveň časté.

2 Klasifikačné stromy

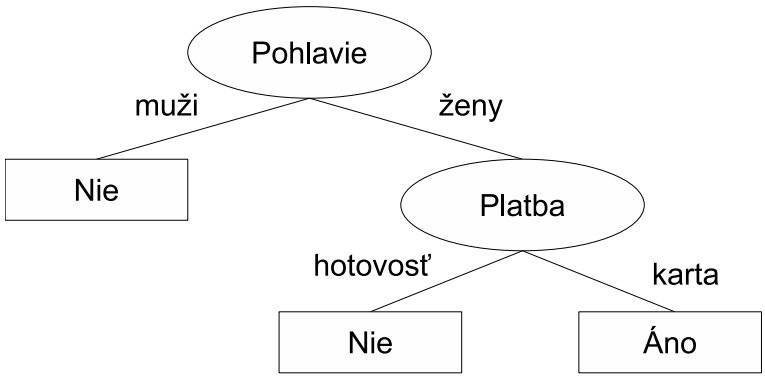
2.1 Pojem a reprezentácia klasifikačného stromu

Klasifikačné stromy (classification trees, niekedy sa nazývajú aj rozhodovacie[2] stromy, decision trees) sú jednou z metód používaných v rámci data miningu na klasifikáciu dát a predikciu. Medzi ich výhody oproti iným používaným modelom (napr. neurónové siete) patrí názornosť a možnosť extrahovať pravidlá, ktoré sú v pozadí modelu (underlying dependencies), ako aj menšia náročnosť na prípravu dát, preto sú v praxi populárne. V tejto kapitole zhrnieme základné prvky algoritmov na budovanie klasifikačných stromov (čerpali sme predovšetkým z publikácii [1], [2]).

Klasifikačný strom je klasifikátor vybudovaný na základe trénovacej množiny, ktorý na základe vstupných údajov predpovedá hodnotu výstupnej (cieľovej) premennej. Formálne, ak máme množinu objektov $O=\{o; o=(o_1,\ldots,o_d)\}$, kde o_i sú hodnoty vstupných vlastností objektu, výstupnú premennú c s oborom hodnôt C (konečná množina v prípade klasifikácie, reálne čísla v prípade predikcie), tak klasifikátorom rozumieme funkciu $f(o): O \rightarrow C$. Našou snahou je zostrojiť takúto funkciu na základe trénovacej množiny $T \subset O$, pričom pre objekty patriace do T už poznáme hodnotu c.

Klasifikačný strom reprezentujeme väčšinou graficky v podobe stromu „rastúceho“ zhora nadol. Jednoduchý strom môže vyzerať napríklad tak ako na obrázku 1 (tu strom predpovedá na základe pohlavia a spôsobu platby, či si zákazník kúpi tovar). Tento príklad vznikol na základe fiktívnej trénovacej množiny dát, ktorej jeden záznam mohol vyzerať napríklad nasledovne:

Muž/Žena	**Karta/Hotovosť**	**Kúpil?**
Žena	Karta	áno



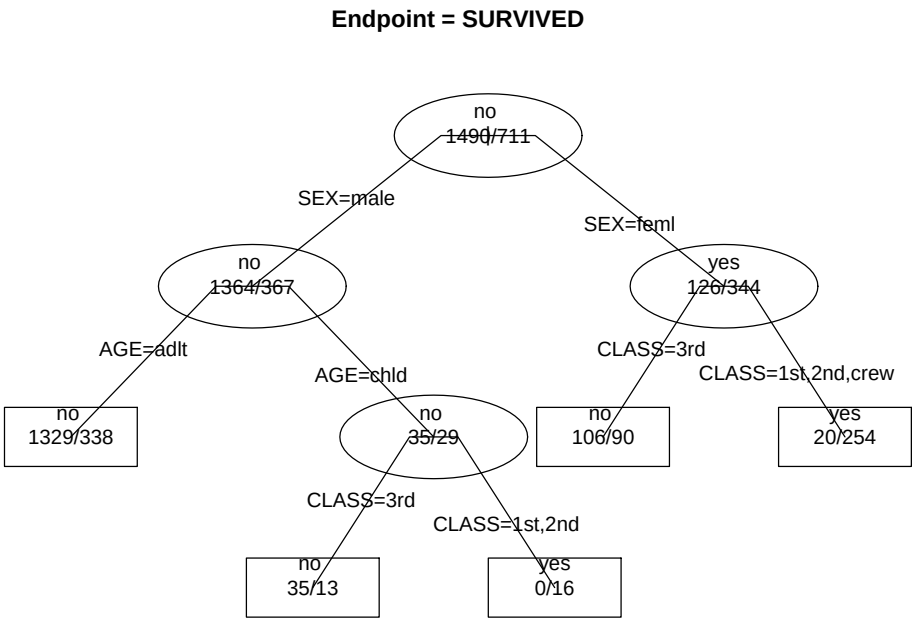
Obrázok 1: Jednoduchý príklad klasifikačného stromu

Keď chceme použiť strom na určenie hodnoty cieľovej premennej pre konkrétny záznam, postupujeme nasledovne: záznam postupne zaraďujeme (zhora nadol) do vetiev stromu podľa rozdeľovacích kritérií stanovených v jednotlivých uzloch. Každý

2 V manažmente má pojem rozhodovacieho stromu odlišný význam – ide o pomôcku používanú na zhodnotenie budúceho vývoja s viacerými alternatívami (napr. na zrátanie čistej súčasnej hodnoty investície, ak poznáme varianty jej možného vývoja, príslušné zisky/straty a pravdepodobnosti).

uzol vlastne predstavuje test, ktorý daný záznam zadelí do práve jednej z uzlu vychádzajúcej vetvy. Na konci skončí záznam v jednom z výstupných uzlov, ktoré predpovedajú hodnotu cieľovej premennej (túto predpoveď však možno spraviť v každom uzle, nielen vo výstupnom).

Nasleduje reálnejší príklad klasifikačného stromu (obrázok 2):



Obrázok 2: Strom predpovedajúci prežitie pasažierov Titanicu

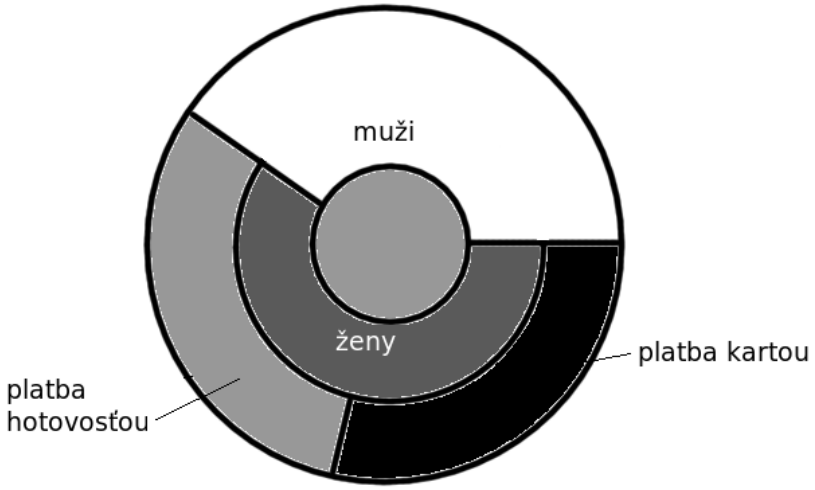
Tento strom je vybudovaný na základe údajov o 2201 pasažieroch Titanicu[3] a na základe pohlavia, triedy a veku sa snaží predpovedať prežitie katastrofy. Strom (vybudovaný pomocou balíka *rpart* v softvéri R) klasifikoval správne 79% záznamov. Ak by sme všetkým pasažierov predpovedali podľa väčšinového výsledku neprežitie (čo zodpovedá úplne najjednoduchšiemu modelu), klasifikovali by sme správne 67,7% záznamov – klasifikačný strom teda v tomto prípade zvýšil presnosť predpovede. Vidíme, že výstupné uzly sú znázornené obdĺžnikom, ostatné elipsou. Zároveň v každom uzle je uvedená predpovedaná hodnota a počet pasažierov ktorí neprežili/prežili stroskotanie. Všimnime si, že výsledný model zodpovedá všeobecnej predstave o udalostiach na Titanicu (najväčšiu šancu na prežitie majú ženy a deti v 1. a 2. triede).

2.1.1 Alternatívne zobrazenie stromu

Iným spôsob grafického zobrazenia stromu na obrázku 1 možno vidieť na obrázku 3. Hlavný uzol predstavuje stred kruhu, a každý prstenec znázorňuje jednu úroveň delenia. V prvom prstenci vidno rozdelenie medzi mužov a ženy, druhý prstenec potom rozdeľuje časť žien podľa spôsobu platby (časť vyjadrujúca mužov sa už ďalej nedelí). Dĺžka oblúka zodpovedá veľkosti jednotlivých vetiev, farba zase hovorí o „hustote“ kladnej hodnoty cieľovej premennej v úseku (teda ak je úsek tmavší, tak medzi záznamami v tomto úseku je viacej tých s cieľovou hodnotou áno – takéto

3 zdroj dát: http://www.cs.toronto.edu/~delve/data/titanic/desc.html

využitie farby je zrejme možné len pre binárnu cieľovú premennú).



Obrázok 3: Alternatívne zobrazenie stromu

2.2 Tvorba stromu

Najdôležitejšími v klasifikačnom strome sú kritériá, podľa ktorých sa rozdeľujú záznamy v jednotlivých uzloch. Algoritmus na budovanie rozhodovacieho stromu začína s celou „trénovacou množinou“ a musí určiť rozdeľovacie kritérium, ktoré čo najlepšie rozdeľuje (triedi) záznamy podľa cieľového atribútu. Na to musí prejsť všetky vstupné premenné, ku každej určiť spôsob rozdelenia na základe tejto premennej a vyhodnotiť kvalitu takého rozdelenia. Spôsob navrhnutého rozdelenia závisí od toho, či je vstupná premenná spojitá alebo diskrétna:

- ak je premenná diskrétna, jedna možnosť je vytvoriť vetvu pre každú nadobúdanú hodnotu, čo však môže viesť k príliš „košatým“ stromom. Druhou možnosťou je zlúčiť spolu tie takto vytvorené vetvy, v ktorých majú záznamy podobné rozdelenie vzhľadom na cieľový atribút (čo sa dá určiť napríklad pomocou štatistického chí-kvadrát testu), alebo vyskúšať všetky rozdelenia do dvoch alebo viacerých skupín.
- ak je premenná spojitá, rozdelenie bude spočívať v určení hranice, ktorá rozdelí záznamy na dve skupiny (s väčšími a menšími hodnotami). Túto hranicu môžeme určiť jednoducho tak, že vyskúšame ako hraničné všetky nadobúdané hodnoty v danej premennej (prípadne, ak je dát veľa, iba ich vzorku), a vyberieme z nich najlepšie rozdelenie.

V prípade, že dáta obsahujú chýbajúce hodnoty, klasifikačné stromy umožňujú jednoducho brať „chýbajúcu hodnotu“ ako ďalšiu prípustnú hodnotu (čo je výhodné, ak samotný fakt, že daný záznam je nekompletný, nie je náhodný, ale podáva nám nejakú informáciu). Niektoré algoritmy (napríklad CART) sa pokúsia zaradiť záznam s chýbajúcou hodnotou na základe zaradenia podobných záznamov.

Kľúčovým je spôsob, ako algoritmus vyhodnotí „kvalitu“ navrhovaného rozdelenia. Na tento účel sa používa niekoľko rôznych meradiel:

- Ak je cieľová premenná diskrétna:
  - Informačná entropia, informačný zisk
  - Giniho index diverzity
  - Chí-kvadrát test
- Ak je cieľová premenná spojitá:
  - pokles v disperzii
  - F-test

Ak vieme vyhodnotiť kvalitu každého rozdelenia, tak si z nich vyberieme to najlepšie, ktoré nám záznamy rozdelí do skupín (vetiev). Rovnakým spôsobom postupujeme ďalej (rekurzívne) v každej z vetiev, pričom pracujeme už iba s dátami, ktoré patria do danej vetvy. Týmto spôsobom algoritmus rozdeľuje dáta, až kým nie je splnené „stopové kritérium“ (závisiace od použitého algoritmu, napr. ak počet záznamov s jednou hodnotou cieľovej premennej prekročí určitú hranicu, napr. 90%, alebo ak počet záznamov vo vetve klesne pod určitú hranicu).

Formálne môžeme zhrnúť algoritmus na budovanie rozhodovacieho stromu nasledovne ([2]):

```
funkcia KonštruujStrom (množina záznamov T)
  ak je splnené stopové kritérium, stop
  inak:
    Pre každý atribút A:
      Pre každé možné rozdelenie na základe A:
        Vyhodnoť kvalitu navrhnutého rozdelenia
    Vyber z navrhnutých rozdelení to najkvalitnejšie
    Pre každú množinu Tᵢ vzniknutú rozdelením:
      KonštruujStrom(Tᵢ)
```

Rozličné konkrétne algoritmy sa potom líšia použitými stopovými kritériami, spôsobom tvorby rozdelení a kritériami na kvalitu rozhodnutia.

2.3 Deliace kritériá

2.3.1 Informačná entropia

Pojem informačnej entropie pochádza z teórie informácie a zaviedol ho Claude Shannon (1947). Informačná entropia diskrétnej náhodnej premennej *X* je definovaná:

$$H(X) = -\sum_{i=1}^{n} p_i \log_2 p_i$$

kde p_i je pravdepodobnosť nastania i-teho stavu. Táto veličina sa dá interpretovať ako priemerný počet bitov potrebný na zakódovanie výsledku realizácie náhodnej premennej. Platí, že entropia nadobúda maximum (rovné $log_2(n)$) ak sú pravdepodobnosti všetkých udalostí rovnaké (teda $p_i=1/n$).Naopak v prípade ak niektorá z udalostí je istá (teda $p_j=1$ pre nejaké j) tak sa entropia rovná nule.

Vidíme teda, že entropia sa dá interpretovať aj ako miera "náhodnosti" alebo "neusporiadanosti" náhodného javu (čo je podobné fyzikálnej definícii entropie). Ak za náhodný jav pokladáme hodnotu cieľovej premennej v našich dátach (za stavy považujeme rôzne nadobúdané hodnoty, a za pravdepodobnosti relatívne početnosti záznamov s danou hodnotou), tak úlohou klasifikačného stromu pri delení dát do podskupín je, aby entropia v týchto skupinách bola menšia (a teda rozdelenie hodnoty cieľového atribútu je "usporiadanejšie"). Za týmto účelom zavádzame veličinu informačný zisk:

$$Gain(X, A) = H(X) - \sum_{i=1}^{n} \frac{|X_i|}{|X|} H(X_i)$$

kde X je množina všetkých záznamov, X_i je množina záznamov v i-tej vetve, A je označenie navrhovaného delenia. Informačný zisk predstavuje rozdiel medzi entropiou pôvodnej množiny dát a váženým priemerom entropií podskupín pri danom delení. Z možných delení v danom uzle stromu potom vyberieme to, ktoré má najväčší informačný zisk.

Nedostatkom tohto kritéria je, že môže preferovať delenie s mnohými vetvami, čo môže byť nežiadúce. Preto sa niekedy používa namiesto informačného zisku jeho modifikovaná podoba nazývaná aj pomerový informačný zisk:

$$\frac{Gain(X, A)}{-\sum_{i=1}^{n} \frac{|X_i|}{|X|} \log_2\left(\frac{|X_i|}{|X|}\right)}$$

Menovateľ predstavuje entropiu samotného rozdelenia (to, kam sa zaradí konkrétny záznam, môžeme považovať za náhodný jav) a teda zložitejšie delenia s väčším počtom vetiev (a teda s vyššou entropiou) sú penalizované väčším menovateľom.

2.3.2 ID3 algoritmus

Jednoduchým algoritmom na budovanie rozhodovacieho stromu, používajúcim ako kritérium informačný zisk, je ID3 (J.R.Quinlan). Predpokladá, že všetky premenné sú diskrétne, pri delení podľa premennej vytvorí novú vetvu pre každú nadobúdanú hodnotu. Algoritmus je jednoduchý – v danom uzle vypočíta informačný zisk pre delenie podľa každej premennej, a vyberie si to najlepšie. V každej novovzniknutej vetve potom postupuje rekurzívne ďalej. Zrejme v ďalších vetvách už nie je pri hľadaní najlepšieho delenia potrebné uvažovať predtým použité premenné (záznamy v aktuálnej vetve už budú nadobúdať len jednu hodnotu v takej premennej). Algoritmus sa zastaví a vytvorí výstupný uzol, ak majú v aktuálnej vetve všetky záznamy rovnakú hodnotu cieľového atribútu, alebo už boli na delenie použité všetky premenné a nie je podľa čoho deliť ďalej (v takom prípade sa určí hodnota výstupného uzla ako prevládajúca hodnota cieľového atribútu), alebo ak je vetva prázdna. Formálne môžeme algoritmus zapísať nasledovne ([3]):

```
funkcia ID3 (R: množina necieľových atribútov,
             C: cieľový atribút,
             S: množina záznamov) vracia rozhodovací strom;
   begin
      Ak S je prázdne, vrát jediný uzol s hodnotou "zlyhanie"
      Ak S pozostáva zo záznamov s rovnakou hodnotou cieľ. atribútu,
```

```
        vrát výstupný uzol s danou hodnotou
    Ak R je prázdna, vrát uzol s hodnotou cieľ. atribútu ktorá je
        najčastejšie medzi záznamami v S
    Nech D je atribút s najväčším Gain(D,S) medzi atribútmi v R;
    Nech {dj| j=1,2, .., m} sú hodnoty atribútu D;
    Nech {Sj| j=1,2, .., m} sú podmnožiny S pozostávajúce zo
        záznamov s hodnotou atribútu D dj
    Vráť strom s uzlom D a vetvami označenými d1,d2,...,dm ktoré
        vedu k stromom:
        ID3(R-{D},C,S1), ID3(R-{D},C,S2), .., ID3(R-{D},C,Sm);
end ID3;
```

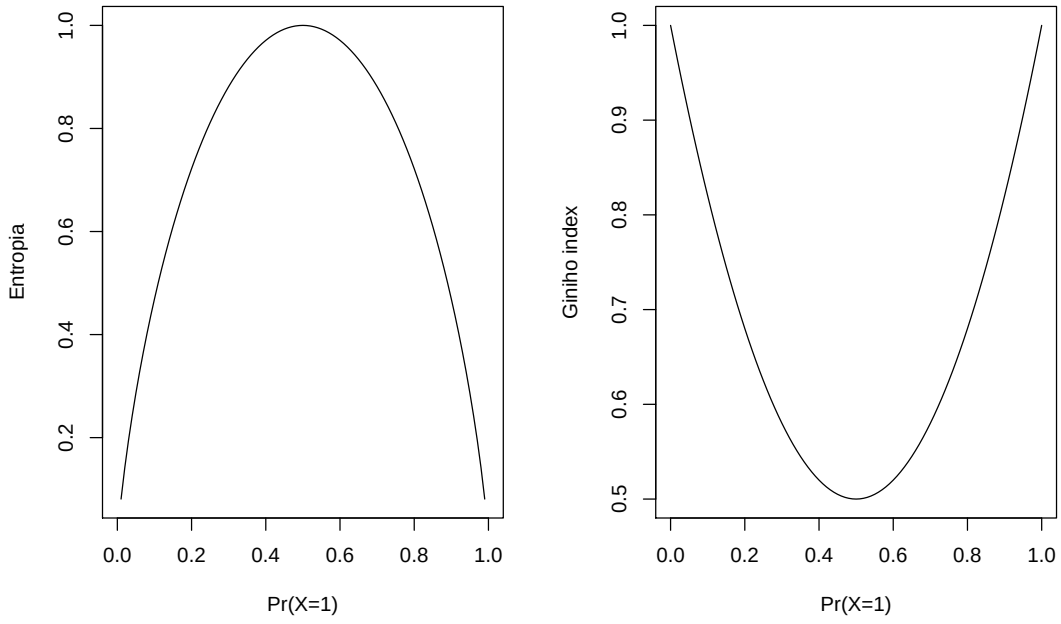
2.3.3 Giniho index diverzity

Inou mierou usporiadanosti záznamov je Giniho miera diverzity, ktorá určuje pravdepodobnosť, že dvojica náhodne vybraných záznamov bude mať rovnakú hodnotu cieľového atribútu. Giniho index teda definujeme nasledovne:

$$G(X)=\sum_{i=1}^{n} p_i^2$$

kde p_i sú relatívne početnosti záznamov s i-tou hodnotou. Giniho index nadobúda maximum ak $p_j=1$ pre nejaké j a minimum (rovné $1/n$) ak sú všetky pravdepodobnosti (početnosti) rovnaké. Pri budovaní stromu potom požadujeme, aby vážený priemer Giniho indexov skupín vytvorených delením bol väčší ako Giniho index celej množiny dát, teda chceme maximalizovať: $\sum_{i}^{n} \frac{|X_i|}{|X|} G(X_i)$

Porovnanie Giniho indexu a informačnej entropie pre náhodnú premennú s alternatívnym rozdelením (nadobúda iba hodnoty 0,1) je na obrázku 4.



Obrázok 4: Informačná entropia a Giniho index pre alternatívne rozdelenie

2.3.4 Chí-kvadrát test

Iný spôsob hodnotenia kvality rozdelenia je pozrieť sa, ako sa zmenilo rozdelenie hodnôt cieľového atribútu vo vzniknutých skupinách. Ak je rozdelenie podobné pôvodnému rozdeleniu v celej množine, tak takéto delenie nám neprináša žiadnu novú informáciu. Preto chceme, aby rozdelenie v skupinách bolo čo najviac odlišné. Na tento účel sa hodí v štatistike používaný χ^2 test, ktorý sa používa na určenie, či namerané hodnoty (realizácie diskrétnej náhodnej premennej) môžu pochádzať z nejakého rozdelenia (nulová hypotéza) alebo nie. Štatistika v tomto teste má podobu $\sum_{j=1}^{m} \frac{(O_j - E_j)^2}{E_j}$, kde *m* je počet hodnôt náhodnej premennej, O_j je pozorovaný počet výsledkov s i-tou hodnotou, a E_j je očakávaný počet zodpovedajúci teoretickému rozdeleniu. Za predpokladu, že platí nulová hypotéza, má štatistika χ^2 rozdelenie s *m-1* stupňami voľnosti. V prípade uzla, v ktorom delíme záznamy do *n vetiev*, môžeme očakávané hodnoty pre i-tu skupinu vyrátať nasledovne (pre *E, O* dolný index označuje hodnotu cieľ. premennej, horný vetvu, v ktorej sa záznam ocitne delením): $E_j^i = \frac{|X_i|}{|X|} q_j$, kde q_j označuje počet záznamov s j-tou hodnotou cieľového atribútu v celej množine, X_i množinu záznamov v i-tej vetve, X celú množinu. Teraz môžeme celkovú štatistiku pre delenie *A* vypočítať (spočítame vlastne hodnotu štatistiky vo všetkých vetvách):

$$C(X, A) = \sum_{i=1}^{n} \sum_{j=1}^{m} \frac{(O_j^i - E_j^i)^2}{E_j^i}$$

Platí, že čím je *C(X,A)* väčšie, tým viac sa rozdelenie hodnôt cieľ. atr. vo vetvách líši od pôvodného, a tým lepšie je dané rozdelenie.

2.3.5 Redukcia disperzie

Ak predpovedáme spojitú cieľovú premennú, rozumnou mierou kvality rozdelenia je pokles disperzie hodnôt vo vetvách delenia (ktoré sú tým pádom „usporiadanejšie“ vzhľadom na cieľovú premennú). Pri výbere delenia teda chceme maximalizovať:

$S(X) - \sum_{i=1}^{n} \frac{|X_i|}{|X|} S(X_i)$, kde *S* je výberová disperzia hodnôt cieľovej premennej.

2.3.6 F test

F test funguje podobne ako χ^2 test, snaží sa určiť, či distribúcie záznamov vo vetvách rozdelenia sú odlišné od distribúcie v pôvodnej množine. Pri spojitej premennej sa na tento účel hodí štatistická metóda ANOVA, ktorá testuje, či sa priemery skupín záznamov navzájom rovnajú (nulová hypotéza) alebo líšia (ak sú odlišné, dané rozdelenie je kvalitné). F-štatistika porovnáva varianciu jednotlivých priemerov s varianciou cieľovej premennej v pôvodnej množine:

$$\frac{\sum_{j=1}^{k}(\bar{y}_j-\bar{y})^2/(k-1)}{\sum_{i=1}^{n}(y_i-\bar{y})^2/(n-k)} \sim F_{k-1,n-k}$$

kde $\bar{y}$ je priemer v celej množine, $\bar{y}_j$ je priemer v j-tej vetve, *k* je počet vetiev, *n* je celkový počet záznamov. Ako vidíme, štatistika má za platnosti nulovej hypotézy Fischerovo rozdelenie s *k-1*, *n-k* stupňami voľnosti (nulovú hypotézu by sme zamietali pre veľké hodnoty štatistiky). Pri hodnotení kvality rozdelenia budeme potom preferovať tie s väčšou hodnotou štatistiky.

2.4 Verifikácia modelu

2.4.1 Osekávanie stromu

Po vybudovaní modelu máme k dispozícii „kompletný“ strom, ktorý však nemusí byť vyhovujúci pre zaraďovanie záznamov mimo trénovacej množiny. Dôvodom je tzv. preučenie (overfitting), čo znamená, že model popisuje až príliš podrobne dáta z trénovacej množiny, pričom zachytené vzťahy nie sú všeobecne platné. Typickou príčinou je príliš málo dát v uzloch na spodných úrovniach stromu, vďaka čomu sa algoritmus vo väčšej miere riadi náhodnými odchýlkami, špecifickými pre trénovaciu množinu.

Niektoré algoritmy sa týmto problémom zaoberajú už pri tvorbe stromu (vetvenie sa zastaví, ak počet záznamov v uzle klesne pod určitú úroveň, alebo ak ani najlepšie vybrané rozdelenie v danom uzle podľa chí-kvadrát testu nevytvára vetvy s odlišnou distribúciou záznamov). Druhá možnosť je osekávanie (prunning), to znamená že po vybudovaní celého stromu sa snažíme vybrať menší podstrom, ktorý má najlepšiu schopnosť predikcie nových záznamov. Každý takýto podstrom zrejme musí obsahovať prvotný (koreňový) uzol a oproti pôvodnému stromu z neho budú odobrané niektoré uzly (vrátane všetkých príslušných nasledujúcich vetiev a uzlov).

Pre osekávanie existuje viacero metód. Pri algoritme CART (classification and regression trees) sa najprv nájde viacero kandidujúcich podstromov s rozličným počtom uzlov. Na výber najvhodnejšieho potrebujeme otestovať každého kandidáta na validačnej množine, t.j. množine dát, ktorú sme nepoužili pri vytváraní stromu. Zmeriame mieru chyby pre každý podstrom a vyberieme kandidáta s najmenšou mierou chyby.

Iný spôsob osekávania, nevyžadujúci validačnú množinu, je použitý v algoritme C5 (ktorý nadväzuje na už spomínaný ID3). Ak chceme odhadnúť chybovosť v nejakom uzle, predpokladáme, že počet chybne zaradených záznamov *E* je náhodná premenná s binomickým rozdelením s parametrami μ (pravdepodobnosť chybného zaradenia, ktorú nepoznáme) a *N* (počet záznamov v uzle). Na základe *N* môžeme vypočítať šírku intervalu spoľahlivosti (na danej hladine významnosti) pre *E* okolo μ (čiže pozorované *E* by sa malo napr. na 95% nachádzať v tomto intervale). Počet chybne zaradených záznamov na trénovacej množine (ktorý je známy) potom pokladáme za dolnú hranicu tohto intervalu (keďže snahou algoritmu je dosiahnuť na trénovacej množine čo najlepšiu klasifikáciu) a hľadaný skutočný počet pesimisticky položíme

rovný hornej hranici intervalu. Na základe toho odhadneme skutočnú mieru chyby v každom uzle stromu. Ak v niektorom uzle bude odhadovaná miera menšia ako v uzloch nachádzajúcich sa v strome vo vetvách pod ním, tak všetky vetvy vychádzajúce z uzla osekneme.

2.4.2 Hodnotenie kvality stromu

Potom, ako sme vybudovali rozhodovací strom, potrebujeme odhadnúť jeho kvalitu, t.j. schopnosť predikcie nových záznamov. Najdôležitejším kritériom pre hodnotenie kvality stromu je už spomínaná miera chybovosti, teda podiel chybne klasifikovaných záznamov (pri predpovedaní diskrétnej premennej). Odhad miery chybovosti sa zvyčajne tvorí na osobitnej množine kompletných dát (testovacia množina), ktoré sme oddelili z trénovacej množiny a nepoužili pri tvorbe stromu. Dôvodom je, že odhad chyby priamo z trénovacej množiny (alebo aj z validačnej množiny, ak sme ju použili na osekávanie) je skreslený a podhodnotený, keďže tieto dáta mali priamy vplyv na vytváranie modelu. Miera chybovosti sa dá vyrátať pre celý strom, aj pre jednotlivé uzly.

Ak strom používame na klasifikáciu (teda predpovedaná premenná je diskrétna), môže nás zaujímať jemnejšie rozdelenie chýb na testovacej množine (napr. chyba pri ktorej chorého pacienta označíme ako zdravého je závažnejšia ako keď je tomu naopak). Na to slúži „matica pomýlenia“ (confusion matrix), ktorej *ij-ty* prvok udáva, koľko záznamov so skutočnou hodnotou *i* bolo klasifikovaných s hodnotou *j*.

Pri predikcii (predpovedaná premenná je spojitá) môžeme pre každý objekt testovacej množiny vyrátať odchýlku medzi predpovedanou a skutočnou hodnotou, čím dostaneme vektor rezíduí. Obvyklou charakteristikou modelu je podobne ako pri lineárnej regresii ich variancia, prípadne štandardná odchýlka.

Ak nie je k dispozícii dostatok dát na vyčlenenie testovacej množiny, môže sa aplikovať m-násobná krížová validácia. Pri nej rozdelíme trénovaciu množinu na *m* skupín, *m-1* skupín použijeme na tvorbu modelu a zvyšnú skupinu na odhad chyby. Toto zopakujeme *m* krát, pričom vždy na odhad chyby vyčleníme inú skupinu. Získame tak *m* rôznych odhadov chyby, ktoré môžeme spriemerovať.

2.4.3 Lift

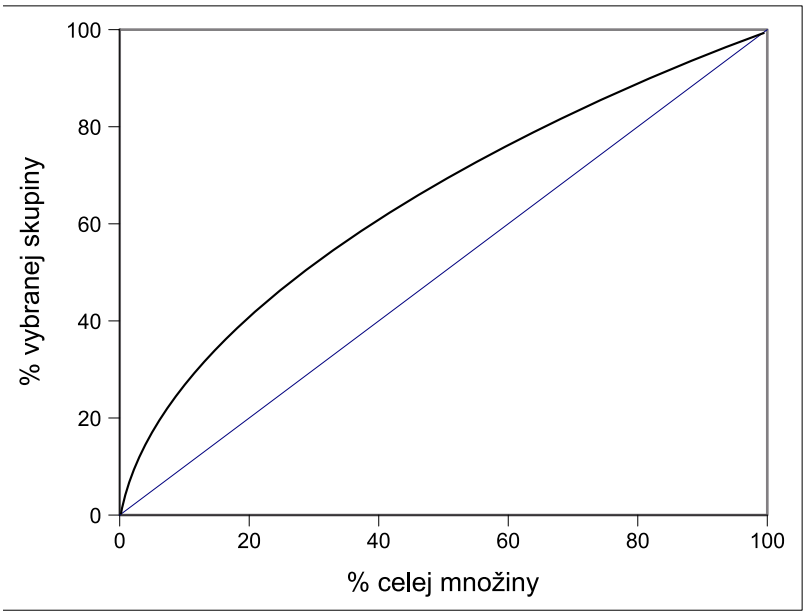
Pomerne rozšírenou mierou kvality modelu pri rôznych modeloch používaných v data miningu na klasifikáciu (diskrétna cieľová premenná) je veličina s názvom lift. Pre danú hodnotu cieľovej premennej a pre danú skupinu dát udáva nárast početnosti záznamov s danou hodnotou v skupine oproti početnosti na celej množine, čo môžeme zapísať nasledovne:

$$Lift(i, A) = \frac{P(\tau(x) = i \mid x \in A)}{P(\tau(x) = i)},$$

i - hodnota cieľovej premennej
A - vybraná skupina dát
$\tau(x)$ - (skutočná) hodnota cieľ. prem. pre záznam x

Často, ak pomocou modelu predpovedáme binárnu premennú (napr. či zákazník bude kladne reagovať na obchodnú ponuku), znázorníme lift na grafe (viď Obrázok 5). Na x-ovej osi je percentil celej množiny zoradenej podľa pravdepodobnosti kladnej odpovede predpovedanej modelom (ide teda o percento všetkých zákazníkov, ktorých oslovíme), na y-ovej osi je percentil množiny záznamov so žiadanou hodnotou v

našom výbere zodpovedajúcom percentilu na x-ovej osi (ide teda o percento zákazníkov, ktorí odpovedia kladne, z tých čo sme si vybrali):



Obrázok 5: Príklad grafu znázorňujúceho lift

Ak by náš model fungoval náhodne, tak zrejme ak by sme oslovili viac zákazníkov, lineárne by stúpol aj počet tých, čo by odpovedali kladne. Ak však máme dobrý model, tento počet by mal na začiatku rásť rýchlejšie (keďže na začiatku x-ovej osi by mala byť časť populácie, ktorej model predpovedal najvyššiu pravdepodobnosť kladnej odpovede). Takéto znázornenie je možné spraviť napr. pre model logistickej regresie, no pre klasifikačné stromy nie je veľmi vhodné.

Prirodzenou interpretáciou pre klasifikačné stromy je vyhodnotiť lift v jednotlivých výstupných uzloch. Zrejme platí, že čím je lift pre niektorý uzol väčší, tým má lepšiu predpovednú schopnosť. Ak sa napríklad v našom príklade s Titanicom zaujímame o pasažierov ktorý neprežili, lift pre uzol dospelých mužov vyrátame ako (1329/1667)/(1490/2201) = 0,797/0,667 = 1,195.

3 CART

CART (classification and regression trees) je algoritmus na tvorbu klasifikačných stromov opísaný v knihe *Classification and Regression trees* od autorov L. Breiman, J.H. Friedman, R.A. Olshen a C.J. Stone (Chapman & Hall/CRC, 1984). My sa budeme zaoberať jeho implementáciou v programe R v knižnici *rpart*, ktorá je popísaná autormi knižnice v publikácii [5] (z ktorej prevažne čerpáme v tejto kapitole). Algoritmus vie narábať s diskrétnymi aj spojitými premennými. Najprv sa budeme zaoberať prípadom diskrétnej cieľovej premennej.

3.1 Označenie

Predpokladáme, že v trénovacej množine máme *n* prvkov, ktoré chceme pomocou stromu klasifikovať do *C* tried. V nasledujúcom texte budeme používať tieto označenia (pomocou ≈ sú označené odhady z dát):

n_i, n_A, n_{iA} počet prvkov i-tej triedy, uzla A, i-tej triedy v uzle A

$\pi_i, i = 1...C$ apriórne pravdepodobnosti tried (ak nie sú známe, tak ich odhadneme ako n_i/n)*

$\tau(x), \tau(A)$ skutočná trieda prvku x, trieda priradená uzlu A

$L(i, j)$ Cena chybnej klasifikácie prvku zo skutočnou hodnotou *i* ako *j*. Ak nie je zadaná, tak $L(i,j)=1$ pre *i* rôzne od *j*, 0 inak (štand. tvar).*

$P(A)$ Pravdepodobnosť, že prvok skončí v uzle A, rovná sa: $\sum_{i=1}^{C} \pi_i P(x \in A | \tau(x) = i) \approx \sum_{i=1}^{C} \pi_i (n_{iA}/n_i) \approx \underbrace{(n_A/n)}_{\text{ak } \pi_i = n_i/n}$

$P(i|A)$ pravdepodobnosť, že prvok patriaci do uzla A má skutočnú triedu *i*:
$= P(\tau(x) = i | x \in A) = \pi_i P(x \in A | \tau(x) = i) / P(x \in A) \approx$
$\approx \pi_i (n_{iA}/n_i) / \sum \pi_i (n_{iA}/n_i) \approx \underbrace{n_{iA}/n_A}_{\text{ak } \pi_i = n_i/n}$

$R(A)$ riziko chybnej klasifikácie v uzle A:
$= \sum_{i=1}^{C} p(i|A) L(i, \tau(A)) \approx \sum_{i=1}^{C} \pi_i L(i, \tau(A)) (n_{iA}/n_i)(n/n_A)$

$R(T)$ riziko chybnej klasifikácie v celom strome: $= \sum_{j=1}^{k} P(A_j) R(A_j)$, kde $A_1...A_k$ sú výstupné uzly stromu

* Novými pojmami sú apriórne pravdepodobnosti a stratová funkcia *L*, ktoré umožňujú prispôsobiť správanie algoritmu tým, že ovplyvňujú výber delenia a výpočet rizika chyby v jednotlivých uzloch. Zadať apriórne pravdepodobnosti je možné napr. ak si myslíme, že rozdelenie cieľovej premennej v trénovacej množine nezodpovedá skutočnému rozdeleniu. Stratová funkcia zase umožňuje zohľadniť rôzne následky chybnej klasifikácie pre nové dáta (napr. pri aplikácii v medicíne neodhalenie choroby je oveľa závažnejšie ako falošne pozitívny výsledok).

3.2 Rozdeľovacie kritériá a tvorba stromu

CART delí záznamy v každom uzle na dve skupiny, pričom uvažuje delenia podľa všetkých vstupných premenných. Pri delení podľa diskrétnej premennej skúša všetky možné rozdelenia jej hodnôt do dvoch skupín, pri spojitej premennej zase hľadá rozdeľujúcu hranicu takisto podobne skúšaním všetkých nadobúdaných hodnôt. Pre nájdenie optimálneho delenia vyhodnocuje jeho kvalitu nasledovne:

Vo všeobecnosti, cieľom rozdelenia uzla A do uzlov A_L, A_R je dosiahnuť čo najväčší pokles „nečistoty“ rozdelenia záznamov s ohľadom na výstupnú premennú:

$$max \quad \Delta I = p(A)\,I(A) - p(A_L)\,I(A_L) - p(A_R)\,I(A_R)$$

kde *I* je miera „nečistoty“ uzla:

$$I(A) = \sum_{i=1}^{C} f\left(P(i|A)\right)$$

pričom ako funkciu *f* používa CART buď Giniho[4] index *f(p)=p(1-p)* alebo informačnú entropiu *f(p)=-p**log(*p*)[5]. Zrejme *I(A)* nadobúda maximum v bode $p_1 = p_2 = ... = p_C = 1/C$.

Zatiaľ sme však nezohľadnili stratovú funkciu *L*. Ak *L* je rôzna od štandardného tvaru (alebo jeho násobku), prispôsobíme tomu apriórne pravdepodobnosti π_i, ktoré túto skutočnosť pri výpočte kvality rozdelenia zohľadnia (altered priors):

Predpokladajme najprv, že matica *L* má tvar:

$$L(i,j) = \begin{cases} L_i & i \neq j \\ 0 & i = j \end{cases} \qquad (1)$$

Potom ak si zadefinujeme upravené apriórne pravdepodobnosti nasledovne:

$$\bar{\pi}_i = \frac{\pi_i L_i}{\Sigma_j \pi_j L_j} \qquad (2)$$

tak platí $\bar{\pi}_i \bar{L}(i,j) = \pi_i L(i,j) \;\; \forall i,j$, kde $\bar{L}$ je násobok štandardnej stratovej matice (zároveň zrejme $\Sigma\, \bar{\pi}_i = 1$). To znamená, že riziko chybnej klasifikácie v uzle, definované ako:

$$R(A) = \sum_{i=1}^{C} L(i,\tau(A))\,P(i|A) = \sum_{i=1}^{C} L(i,\tau(A))\,\pi_i\,P(x \in A|\tau(x)=i)/P(x \in A) \approx$$

$$\approx \sum_{i=1}^{C} \underline{L(i,\tau(A))\,\pi_i}\,(n_{iA}/n_i)(n/n_A)$$

sa nezmení pri použití $\bar{\pi}_i$, $\bar{L}$ vo výpočtoch. Keďže $\bar{L}$ je násobkom štand. tvaru stratovej matice, môžeme v takomto prípade použiť predchádzajúce vzťahy pre výpočet kvality rozdelenia (ale pri výpočte rizika chyby v uzle neskôr použijeme pôvodné π_i). Ak matica *L* nie je v tvare (1) (čo zrejme môže nastať len pre *C*>2), tak vo vzťahu (2) použijeme odhad $L_i = \Sigma_j L(i,j)$.

Zaujímavou vlastnosťou upravených apriórnych pravdepodobností je, že posúvajú

4 táto definícia sa líši od Giniho indexu v časti 2.3.3 svojimi vlastnosťami, podobne ako pri entropii tu platí f(0)=f(1)=0, maximum je pre p=0.5

5 na rozdiel od definície v časti 2.3.1 je tu použitý prirodzený logaritmus, čo je však iba malý rozdiel

maximum miery nečistoty uzla do bodu, ktorý zodpovedá pomeru strát pri chybnej klasifikácii pre jednotlivé triedy.

Algoritmus rekurzívne rozdeľuje dáta v zmysle všeobecného postupu popísaného v kapitole 2, zastaví sa, keď počet záznamov v uzle klesne pod stanovenú hranicu. Predpovedaná hodnota v každom uzle sa určí ako prevládajúca hodnota cieľovej premennej spomedzi záznamov v danom uzle.

3.3 Osekávanie stromu a krížová validácia

Potom, ako s pomocou vyššie opísaných rozdeľovacích kritérii vybudujeme celý strom, snažíme sa z neho vybrať redukovaný model, ktorý bude čo najlepšie klasifikovať nové dáta. Označme *T* strom, |*T*| počet jeho koncových uzlov a zaveďme parameter komplexnosti α>0. Definujme upravené riziko chyby, ktoré penalizuje väčší počet výstupných uzlov:

$$R_\alpha(T) = R(T) + \alpha|T|$$

Pre každé α>0 vieme určiť T_α ako podstrom, ktorý má pre dané α minimálnu R_α. Zrejme platí že T_0 je celý strom, T_∞ bude pozostávať iba zo začiatočného uzla a ak α>β tak $T_\alpha \subseteq T_\beta$. Keďže všetkých rozličných podstromov je konečný počet, môžeme určiť postupnosť kritických hodnôt *0*, α_1, α_2, …, α_{m-1}, ∞ tak, že na intervale medzi týmito hodnotami $\langle 0,\alpha_1\rangle, (\alpha_1,\alpha_2\rangle, \ldots, (\alpha_{m-1},\infty)$ bude príslušný T_α rovnaký. Následne si vyrátame čísla $\beta_1=0, \beta_2=\sqrt{\alpha_1\alpha_2}$, ... , $\beta_{m-1}=\sqrt{\alpha_{m-2}\alpha_{m-1}}, \beta_m=\infty$, ktoré budú ležať (okrem prvého a posledného) vnútri uvedených intervalov.

Teraz môžeme pristúpiť k samotnej krížovej validácii. Rozdelíme trénovaciu množinu na *m* skupín (typicky *m*=10), jednu vynecháme a na ostatných vybudujeme celý strom a podstromy $T_{\beta_1}, T_{\beta_2}, \ldots, T_{\beta_m}$, na ktorých vyrátame riziko chyby na vynechanej časti dát. Toto opakujeme *m* krát, zakaždým vynechajúc inú skupinu dát. Nakoniec pre každé β_i vyrátame priemernú chybu a jej štandardnú odchýlku.

Nakoniec vyberieme to β_i, ktoré má najmenší odhad chyby. Alternatívne môžeme z tých β_i, ktorých chyby sa nachádzajú v okruhu jednej štand. odchýlky od minima, vybrať to s najmenším zodpovedajúcim podstromom. Výsledný strom je potom podstrom zodpovedajúci vybranému β_i vybudovaný nad celou trénovacou množinou.

3.4 Chýbajúce dáta

Ak pre niektorý záznam nie sú k dispozícii všetky vstupné premenné, algoritmus sa ho pokúsi napriek tomu využiť pri budovaní stromu a pri klasifikácii (zrejme však každý objekt trénovacej množiny musí mať hodnotu predpovedanej premennej). Pri hodnotení kvality delenia podľa premennej, v ktorej sa vyskytujú záznamy s chýbajúcou hodnotou, sa tieto jednoducho neberú do úvahy (a členy $P(A_L)$. $P(A_R)$ sú preškálované, aby mali súčet 1).

Zložitejšia situácia nastáva, keď už je delenie zvolené a máme prvok zaradiť do jednej z vetiev práve podľa premennej, v ktorej mu chýba hodnota. Pre taký prípad CART v každom uzle definuje aj náhradné kritériá, podľa ktorých sa takýto záznam zaradí. Tieto sa hľadajú nasledovným postupom:

1. Pre každú z ostatných vstupných premenných skonštruujeme pomocný strom s

iba jedným delením podľa tejto premennej (po prvom delení nepokračujeme rekurzívne ďalej, straty a apriórne pravdepodobnosti sú štandardné), pričom cieľovou premennou je práve binárne rozdelenie v uzle, ktoré chceme nahradiť. Ak napríklad hľadáme náhradné rozdelenie pre kritérium veku (<40 rokov, >40 rokov), cieľovou premennou bude to, v ktorej z týchto dvoch skupín skončí záznam (zrejme inkriminované prvky, u ktorých táto hodnota chýba a kvôli ktorým hľadáme náhradné kritériá, v tomto postupe vynecháme).

2. Pre každé z týchto delení vyhodnotíme chybovosť, rovnako aj pre jednoduché kritérium „choď s väčšinou“, ktoré zaradí záznam do vetvy s väčším počtom ostatných záznamov.
3. Zoradíme delenia podľa chybovosti (ignorujeme tie, ktorých chyba je horšia ako pravidlo „choď s väčšinou“), čím dostaneme zoznam náhradných kritérií.

Záznam, ktorý nemožno zaradiť kvôli chýbajúcej hodnote podľa primárneho kritéria, potom zaradíme podľa prvého náhradného kritéria, prípadne podľa druhého (ak v prvom náhradnom tiež prekáža chýbajúca hodnota), tretieho atď...

3.5 Spojitá cieľová premenná

Klasifikačné stromy môžu byť použité aj pre predikciu spojitej premennej. Za týmto účelom prispôsobíme algoritmus nasledovne:

- Kritériom na voľbu optimálneho rozdelenia je rozdiel medzi sumami štvorcov odchýlok, presnejšie $\max \ \Delta I = SS_A - (SS_{A_L} + SS_{A_R})$, kde $SS_A = \sum_{y_i \in A} (y_i - \bar{y})$ (čo je ekvivalentné kritériu redukcie disperzie).
- Hodnota uzla sa určí ako priemer hodnôt cieľovej premennej záznamov v uzle.
- Chyba uzla bude štandardná odchýlka hodnôt cieľovej premennej v uzle.
- Pre celý strom môžeme vypočítať relatívnu chybu podobne ako pri lineárnej regresii $1\text{-}R^2$.

4 Príklady

Po teoretickom opise algoritmu CART uvedieme niekoľko príkladov, ktoré demonštrujeme s pomocou balíka *rpart* v štatistickom (voľne šíriteľnom) prostredí R[6] vrátane uvedenia základných príkazov na ovládanie balíka[7] (predpokladáme základnú znalosť prostredia a príkazov v R). V príkladoch budeme používať pôvodné apriórne pravdepodobnosti a straty (keďže pre zvolené dáta nemáme dôvod ich meniť). Pri prvých dvoch príkladoch (v ktorých je predpovedaná premenná diskrétna), sme vyskúšali použiť obe deliace kritériá (Giniho index aj informačnú entropiu), výsledok sa však nezmenil.

4.1 Titanic

Vrátime sa znova k príkladu pasažierov Titanicu spomenutému v časti 2.1. Dataset obsahuje 2201 pozorovaní so štyrmi (kategorickými) premennými:

CLASS	trieda, ktorou pasažier cestoval (hodnoty *1st, 2nd, 3rd, crew*)
AGE	vek pasažiera (hodnoty *adult, child*)
SEX	pohlavie pasažiera (hodnoty *male, female*)
SURVIVED	prežitie pasažiera (hodnoty *yes, no*)

Ak máme dáta načítané v data.frame s názvom titanic, strom vytvoríme nasledovne:

```
> library(rpart) #nacitanie kniznice
> strom <- rpart(formula=SURVIVED ~ CLASS + AGE + SEX, data=titanic)
> strom
n= 2201

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 2201 711 no (0.6769650 0.3230350)
   2) SEX=male 1731 367 no (0.7879838 0.2120162)
     4) AGE=adult 1667 338 no (0.7972406 0.2027594) *
     5) AGE=child 64  29 no (0.5468750 0.4531250)
      10) CLASS=3rd 48  13 no (0.7291667 0.2708333) *
      11) CLASS=1st,2nd 16   0 yes (0.0000000 1.0000000) *
   3) SEX=female 470 126 yes (0.2680851 0.7319149)
     6) CLASS=3rd 196  90 no (0.5408163 0.4591837) *
     7) CLASS=1st,2nd,crew 274  20 yes (0.0729927 0.9270073) *
```

Vo výstupe vidíme zoznam do seba vnorených uzlov, ku každému je uvedené kritérium, podľa ktorého boli doňho zaradené záznamy, počet záznamov, strata (rovnajúca sa $\Sigma_{x \in A} L(\tau(A), \tau(x))$, pri štandardnej stratovej funkcii sa rovná počtu zle klasifikovaných záznamov), predpovedaná hodnota a pravdepodobnosti *P(i|A)*.
Je možné nechať si zobraziť aj výstup z krížovej validácie:

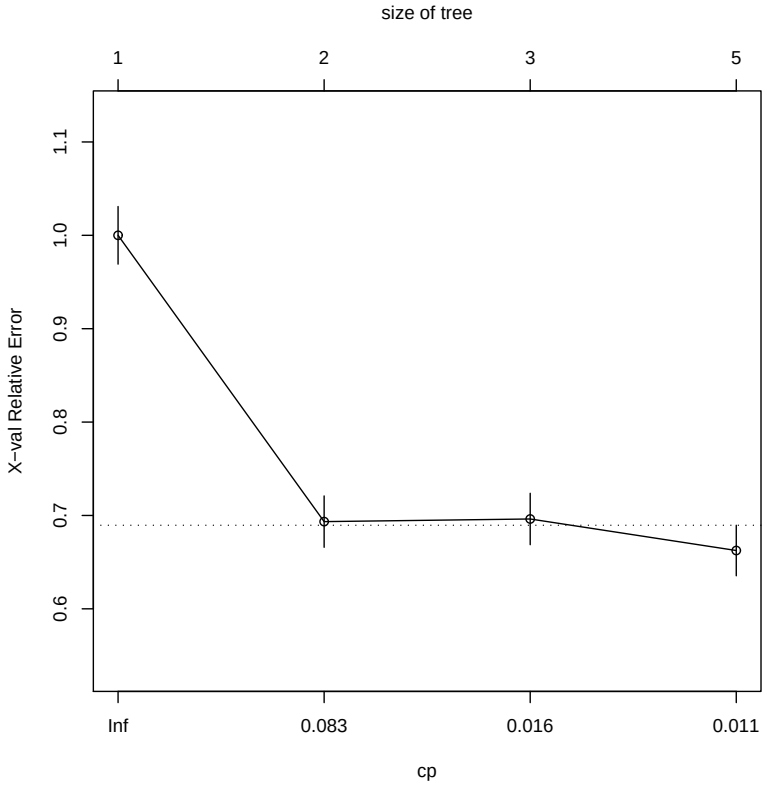
---

6 R je voľne šíriteľnou implementáciou jazyka S určeného na štatistické výpočty a v základných a doplnkových knižniciach obsahuje veľké množstvo funkcií a metód, vrátane tých používaných pri data miningu. Inštalácia pre rôzne systémy je dostupná na stiahnutie na stránke http://www.r-project.org/, balík rpart sa dá nainštalovať príkazom `install.packages('rpart')`

7 Dokumentácia k jednotlivým príkazom sa dá po načítaní knižnice vyvolať príkazom `help(prikaz)`, zoznam príkazov je možné zobraziť pomocou `help(package=rpart)`

```
> printcp(strom)
Classification tree:
rpart(formula = SURVIVED ~ CLASS + AGE + SEX, data = titanic)

Variables actually used in tree construction:
[1] AGE   CLASS SEX

Root node error: 711/2201 = 0.32303
n= 2201

        CP nsplit rel error  xerror     xstd
1 0.306610      0   1.00000 1.00000 0.030857
2 0.022504      1   0.69339 0.69339 0.027510
3 0.011252      2   0.67089 0.69902 0.027589
4 0.010000      4   0.64838 0.66385 0.027083
```

V tabuľke na konci výstupu sú uvedené jednotlivé kandidátske podstromy. Stĺpec CP uvádza hraničné hodnoty parametra komplexnosti α_i, stĺpec nsplit udáva počet rozdelení (počet koncových uzlov mínus 1) optimálneho podstromu na danom intervale α (čiže napríklad strom s dvomi rozdeleniami bude optimálny pre $\alpha \in (0.011252\ ,\ 0.022504)$). Stĺpec rel.error udáva chybu daného podstromu na celej trénovacej množine (preškálovanú aby chyba zač, uzla bola 1), stĺpce xerror, xstd udávajú odhad chyby z krížovej validácie a jej štand. odchýlku (podobne preškálovanú). Program umožňuje aj grafické znázornenie (`plotcp(strom)`), v ktorom vidno aj hranicu jednej štand. odchýlky od min. odhadnutej chyby (Obrázok 6):



Obrázok 6: Grafický výstup z krížovej validácie pre dáta Titanic

Na základe tabuľky aj podľa pravidla o jednej štand. odchýlke je najlepším

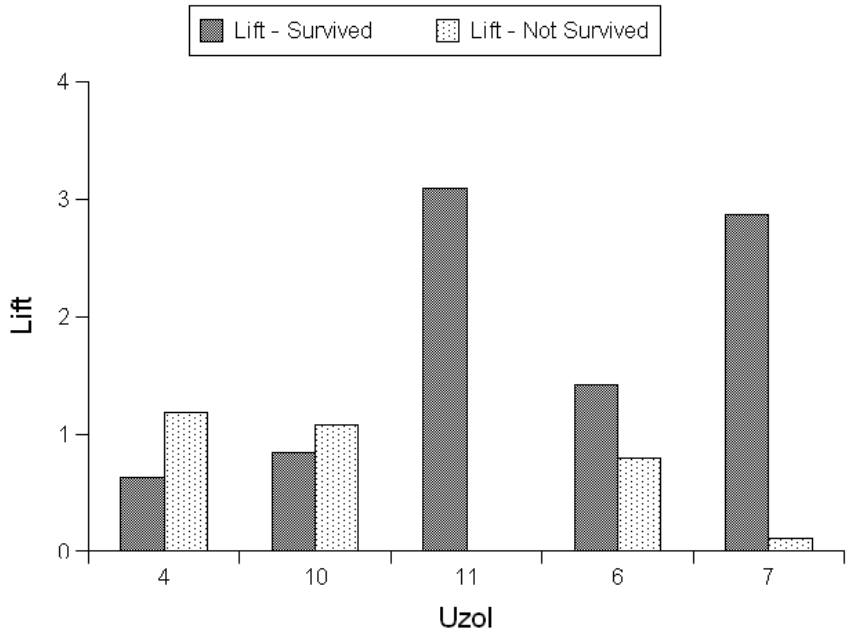
podstromom ten so štyrmi deleniami, ktorý je graficky znázornený na obrázku 2. (príkaz `post(strom, filename="")`).

Detailný opis stromu môžeme vyvolať príkazom `summary(strom)`, ktorý nám okrem predchádzajúcich výstupov ponúkne aj detailný popis jednotlivých uzlov, príklad popisu jedného uzla:

```
Node number 3: 470 observations,    complexity param=0.02250352
  predicted class=yes  expected loss=0.2680851
    class counts:   126   344
   probabilities: 0.268 0.732
  left son=6 (196 obs) right son=7 (274 obs)
  Primary splits:
      CLASS splits as  RRLR, improve=50.015320, (0 missing)
      AGE   splits as  RL,   improve= 1.197586, (0 missing)
  Surrogate splits:
      AGE splits as  RL, agree=0.619, adj=0.087, (0 split)
```

Za zmienku tu stojí položka complexity parameter, ktorá udáva hraničnú hodnotu parametra komplexnosti α pre ktorú bude tento uzol patriť kandidujúcemu podstromu. V časti Primary splits sú uvedené uvažované delenia (improvement je počet dát krát miera zmeny nečistoty), Surrogate splits uvádza alternatívne kritériá pre záznamy s chýbajúcou hodnotou (ako však vidíme, v tomto prípade sa žiadne takéto záznamy nevyskytli).

Rpart neumožňuje vypočítať lift pre jednotlivé uzly, čo nám však nebráni tieto vyrátať manuálne. Výsledky sú zhrnuté v nasledujúcom grafe na obrázku 7. Najväčší lift sa v tomto prípade dosahuje v uzloch 11 a 7.



Obrázok 7: Lift pre jednotlivé uzly v strome Titanic

4.2 Iris

V ďalšom príklade sa budeme zaoberať dátami o kvetoch kosatcov[8]:

8 zdroj dát: dataset iris, knižnica datasetov pre strojové učenie, University of California, Irvine: ftp://ftp.ics.uci.edu/pub/machine-learning-databases/

Sepal.length	dĺžka kalištného lístka
Sepal.width	šírka kalištného lístka
Petal.length	dĺžka okvetného lístka
Petal.width	šírka okvetného lístka
Species	druh (hodnoty *setosa/versicolor/virginica*)

Dataset obsahuje 150 riadkov, budeme sa snažiť predpovedať druh kvetu. Pokúsime sa pritom vytvoriť čo najpresnejší model a preto znefunkčníme pôvodné stop. kritéria algoritmu (napr. min. veľkosť uzla pre rozdelenie je pôvodne nastavená na 20, my ju nastavíme na 2). Výsledkom je rozsiahly strom s 9 výstupnými uzlami, ktorý správne klasifikuje každý záznam, zároveň však obsahuje niekoľko malých uzlov obsahujúcich iba pár záznamov:

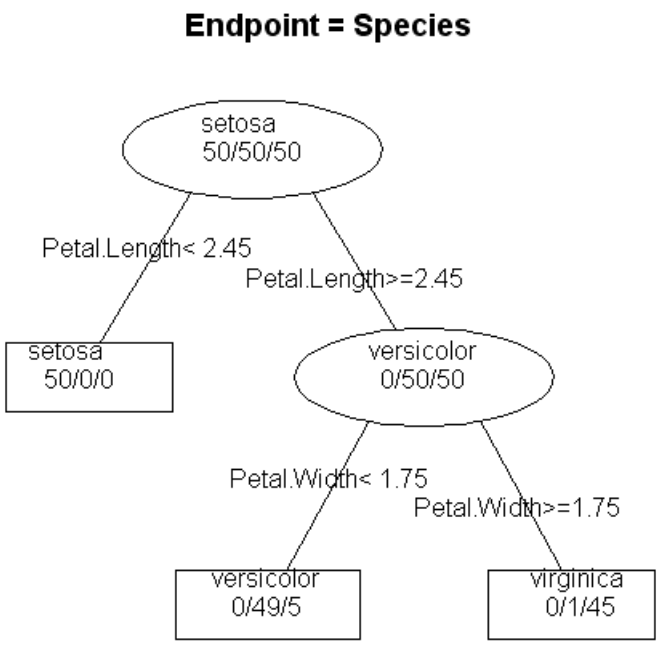
```
n= 150

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 150 100 setosa (0.33333333 0.33333333 0.33333333)
   2) Petal.Length< 2.45 50   0 setosa (1.00000000 0.00000000 0.00000000) *
   3) Petal.Length>=2.45 100  50 versicolor (0.00000000 0.50000000 0.50000000)
     6) Petal.Width< 1.75 54   5 versicolor (0.00000000 0.90740741 0.09259259)
      12) Petal.Length< 4.95 48   1 versicolor (0.00000000 0.97916667 0.02083333)
        24) Petal.Width< 1.65 47   0 versicolor (0.00000000 1.00000000 0.00000000) *
        25) Petal.Width>=1.65 1   0 virginica (0.00000000 0.00000000 1.00000000) *
      13) Petal.Length>=4.95 6   2 virginica (0.00000000 0.33333333 0.66666667)
        26) Petal.Width>=1.55 3   1 versicolor (0.00000000 0.66666667 0.33333333)
          52) Sepal.Length< 6.95 2   0 versicolor (0.00000000 1.00000000 0.00000000) *
          53) Sepal.Length>=6.95 1   0 virginica (0.00000000 0.00000000 1.00000000) *
        27) Petal.Width< 1.55 3   0 virginica (0.00000000 0.00000000 1.00000000) *
     7) Petal.Width>=1.75 46   1 virginica (0.00000000 0.02173913 0.97826087)
      14) Petal.Length< 4.85 3   1 virginica (0.00000000 0.33333333 0.66666667)
        28) Sepal.Length< 5.95 1   0 versicolor (0.00000000 1.00000000 0.00000000) *
        29) Sepal.Length>=5.95 2   0 virginica (0.00000000 0.00000000 1.00000000) *
      15) Petal.Length>=4.85 43   0 virginica (0.00000000 0.00000000 1.00000000) *
```

Takýto model nie je vhodný pre predpovedanie druhu nových kvetov, keďže pravidlá v týchto malých uzloch môžu odrážať iba náhodné osobitosti konkrétnych prvkov trénovacej množiny. Výstup z krížovej validácie:

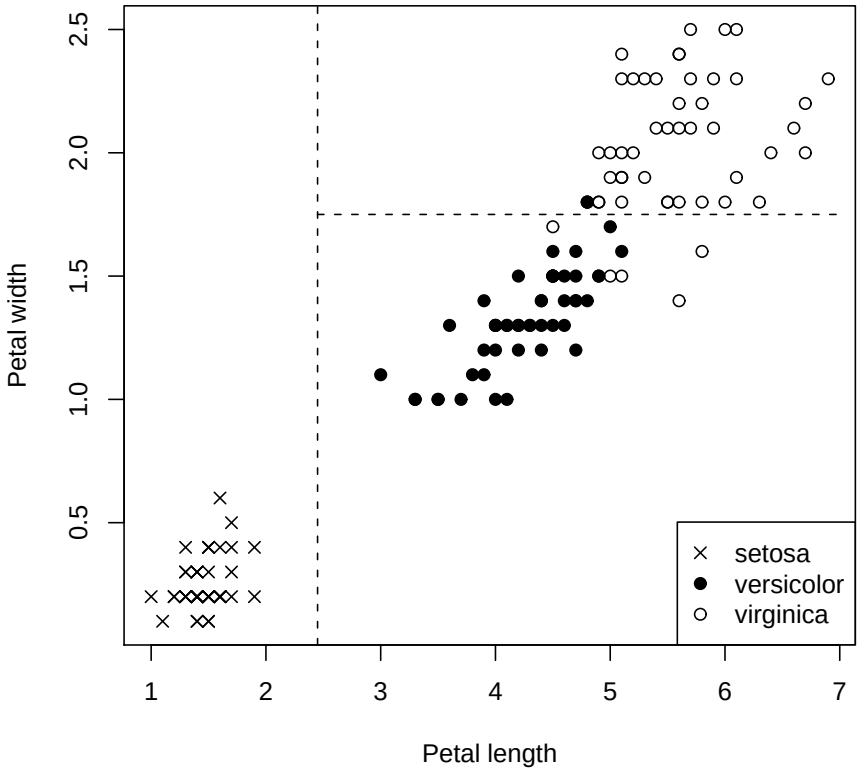
```
     CP nsplit rel error xerror     xstd
1 0.500      0      1.00   1.19 0.049592
2 0.440      1      0.50   0.67 0.060888
3 0.020      2      0.06   0.10 0.030551
4 0.010      3      0.04   0.10 0.030551
5 0.005      6      0.01   0.09 0.029086
6 0.000      8      0.00   0.09 0.029086
```

Vidíme, že posledné štyri možnosti majú skoro rovnaký odhad chyby, preto je správne v tomto prípade zvoliť ako model podstrom s dvomi deleniami (Obrázok 8).

Je zrejmé, že aj tento jednoduchý model má vysokú úspešnosť (chybne klasifikuje iba 6 prípadov) a pritom netrpí spomínanými neduhmi. Keďže pri konštrukcii stromu boli využité iba dve premenné, môžeme si kvôli väčšej názornosti znázorniť dvojrozmerné rozdelenie (každý bod predstavuje jeden kvet, rôzne druhy sú znázornené rôznymi značkami, deliace kritériá sú naznačené pruhovanými čiarami), ktoré udáva výsledný strom (Obrázok 9). Na grafe môžeme vidieť, že strom úspešne oddelil kvety druhy setosa od ostatných, a potom sa snažil čo najlepšie stanoviť hranicu medzi druhmi versicolor a virginica, ktoré medzi sebou plynule prechádzajú.



Obrázok 8: Klasifikačný strom pre dáta iris

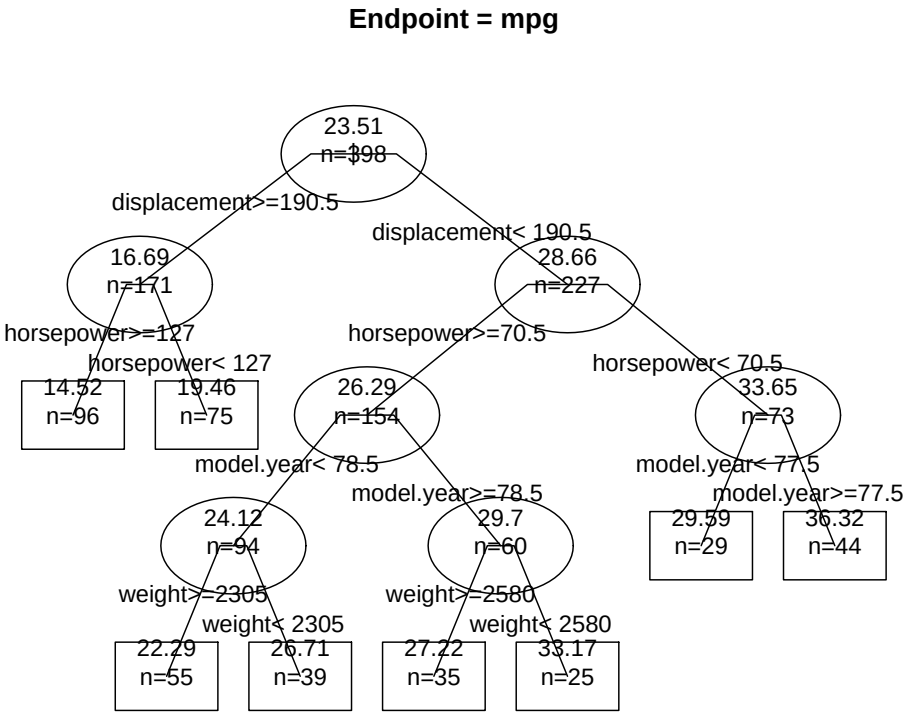


Obrázok 9: Dvojrozmerné znázornenie stromu pre dáta iris

4.3 Spotreba paliva áut

V poslednom príklade sa pokúsime predpovedať spojitú premennú, spotrebu áut[9]. Vo výslednom strome vystupujú nasledujúce premenné (medzi ďalšími premennými, nepoužitými vo výslednom modele, boli počet valcov, zrýchlenie a pôvod značky auta):

displacement	objem valcov
weight	váha auta
model.year	rok výroby
horsepower	sila motora v konských silách
mpg	spotreba v míľach na galón



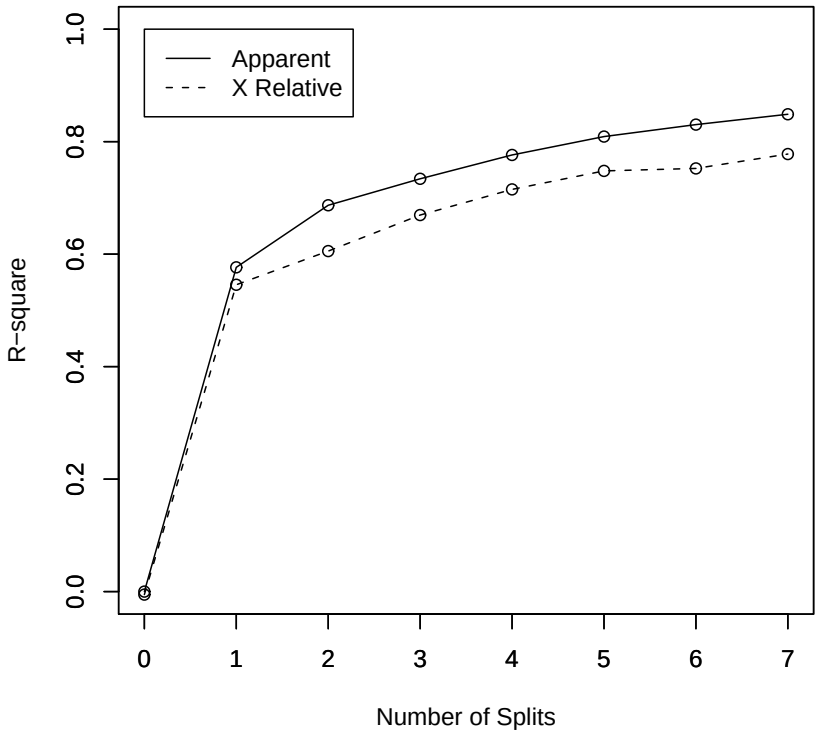
Obrázok 10: Klasifikačný strom pre dáta auto-mpg

Predpovedanou hodnotou (prvý riadok v uzle, druhý riadok je počet záznamov) v uzle je priemerná hodnota spotreby spomedzi záznamov v uzle (pri textovom výstupe by bola uvedená aj disperzia pre každý uzol). Pri pohľade na strom je vidno, že lepšiu spotrebu (viac míľ na galón paliva) majú vozidlá s menším objemom valcov, s menšou silou motora, s menšou váhou a s novším rokom výroby.

V rámci výstupu z krížovej validácie si môžeme okrem závislosti relatívnej chyby ako v predch, príkladoch, zobraziť aj závislosť koeficientu determinácie R^2 od počtu delení v strome (Obrázok 11). R^2 môžeme definovať rovnako ako pri lineárnej regresii (y_i – hodnoty cieľ. prem. , $\hat{y}_i$ – predpovedané hodnoty , $\bar{y} = (1/n)\Sigma\, y_i$):

9 zdroj dát: dataset auto-mpg, knižnica datasetov pre strojové učenie, University of California, Irvine ftp://ftp.ics.uci.edu/pub/machine-learning-databases/

$$R^2 = 1 - RSS/TSS, \quad RSS = \Sigma (y_i - \hat{y}_i)^2, \quad TSS = \Sigma (y_i - \bar{y})^2$$



Obrázok 11: Závislosť R^2 od počtu rozdelení (plná čiara – výsledok na celej množine, pruhovaná čiara – odhad z krížovej validácie)

Náš model dosahuje R^2 takmer 0,8, čo sa väčšinou pokladá za dobrý model. Zaujímavé môže byť porovnanie s obyčajnou lineárnou regresiou aplikovanou na tie isté dáta (použijeme iba tie premenné, ktoré sa objavili aj v klasifikačnom strome):

```
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  -1.361e+01  4.198e+00  -3.242  0.00129 **
displacement  1.839e-03  5.356e-03   0.343  0.73144
horsepower   -6.694e-03  1.066e-02  -0.628  0.53040
weight       -6.590e-03  5.832e-04 -11.300  < 2e-16 ***
model.year    7.505e-01  5.244e-02  14.312  < 2e-16 ***
---
Residual standard error: 3.434 on 387 degrees of freedom
  (6 observations deleted due to missingness)
Multiple R-Squared: 0.8084,     Adjusted R-squared: 0.8064
F-statistic: 408.1 on 4 and 387 DF,  p-value: < 2.2e-16
```

Z výsledkov regresie je zrejmé, že koeficienty determinácie sú v oboch modeloch podobné, a teda pre tieto dáta je model založený na klasifikačnom strome zmysluplný. Môžeme si ešte všimnúť, že pri lineárnej regresii došlo k vyradeniu šiestich záznamov kvôli chýbajúcim dátam, zatiaľ čo v klasifikačnom strome boli zaradené pomocou náhradných kritérií (čo by sa ukázalo príkazom `summary`).

Záver

Cieľom našej práce bolo podať stručný úvod do problematiky klasifikačných stromov, ktoré patria medzi hlavné metódy používané v procese data miningu. Zadefinovali sme pojem klasifikačného stromu a načrtli všeobecnú schému algoritmu na jeho vybudovanie pomocou trénovacej množiny. Následne sme sa venovali detailnejšie jednotlivým prvkom algoritmu, hlavne rôznym používaným deliacim kritériám a opísali sme fázu verifikácie, úpravy a hodnotenia výsledného modelu. S pomocou týchto všeobecných znalostí sme v ďalšej časti práce podrobnejšie rozobrali konkrétny algoritmus na tvorbu klasifikačných stromov, CART.

Prebranú teóriu sme prakticky ilustrovali niekoľkými príkladmi s použitím reálnych dát a voľne šíriteľného softwéru. V dvoch prípadoch sme strom použili na klasifikáciu – v prípade pasažierov Titanicu sme získali model potvrdzujúci všeobecnú predstavú o spomínaných udalostiach, pri klasifikácii kvetov kosatcov sme zase dostali úspešný, no pritom intuiívne jednoduchý model. V poslednom príklade sme použili strom na predpovedanie spojitej premennej a porovnali sme výsledný model s metódou lineárnej regresie, pričom pre tieto dáta boli oba modely porovnateľne úspešné.

Literatúra

1. Michael J. A. Berry, Gordon S. Linoff. *Data Mining Techniques.* 2nd ed. Wiley Publishing, 2004. ISBN 0-471-47064-3
2. J. Paralič. *Objavovanie znalostí v databázach*. Košice : Elfa, 2003. ISBN 80-89066-60-7. Dostupné tiež: http://www.tuke.sk/paralicj/prezentacie/OZ/ObjavovanieZnalostivDB.pdf
3. Giorgio Ingargio. *Building Classification Models: ID3 and C4.5* [online]. Dostupné na: http://www.cis.temple.edu/~ingargio/cis587/readings/id3-c45.html [cit. 4.6.2007]
4. Kurt Thearling. *An Overview of Data Mining Techniques* [online]. Dostupné na: http://www.thearling.com/text/dmtechniques/dmtechniques.htm [cit. 4.6.2007]
5. Terry M. Therneau, Elizabeth J. Atkinson. *An Introduction to Recursive Partitioning Using the RPART Routines* [online]. 1997. Dostupné na: http://www.mayo.edu/hsr/techrpt/61.pdf [cit. 4.6.2007]