Drzewa klasyfikacyjne

Konspekt do zajęć: Statystyczne metody analizy danych

Agnieszka Nowak-Brzezińska

11 stycznia 2010

1 Wprowadzenie

Drzewa klasyfikacyjne [1] jako reprezentacja wiedzy o klasyfikacji są dość atrakcyjne i popularność, jaką cieszą się wykorzystujące je algorytmy uczenia się pojęć, jest uzasadniona ich istotnymi zaletami. Po pierwsze, mogą one reprezentować dowolnie złożone pojęcia pojedyncze i wielokrotne, jeśli tylko ich definicje można wyrazić w zależności od atrybutów używanych do opisu przykładów. Mówiąc bardziej precyzyjnie, za pomocą drzewa, może być reprezentowana dowolna funkcja odwzorowująca wartości wszystkich określonych na dziedzinie atrybutów na zbiór kategorii, czyli dowolna dopuszczalna hipoteza. Reprezentacja wykorzystująca drzewa, jest przy tym, dla typowych pojęć, dość efektywna pamięciowo, a także, co na pewno zasługuje na uwagę, drzewa takie umożliwiają niezwykle efektywną implementację procesu klasyfikowania przykładów. Ponadto istnieje łatwe przejście od drzewa do reprezentacji regułowej uważanej przez wielu, za najbardziej czytelną.

Cel zajęć

Celem zajęć jest poznanie metod budowy i analizy drzew klasyfikacyjnych przy użyciu środowiska *R*.

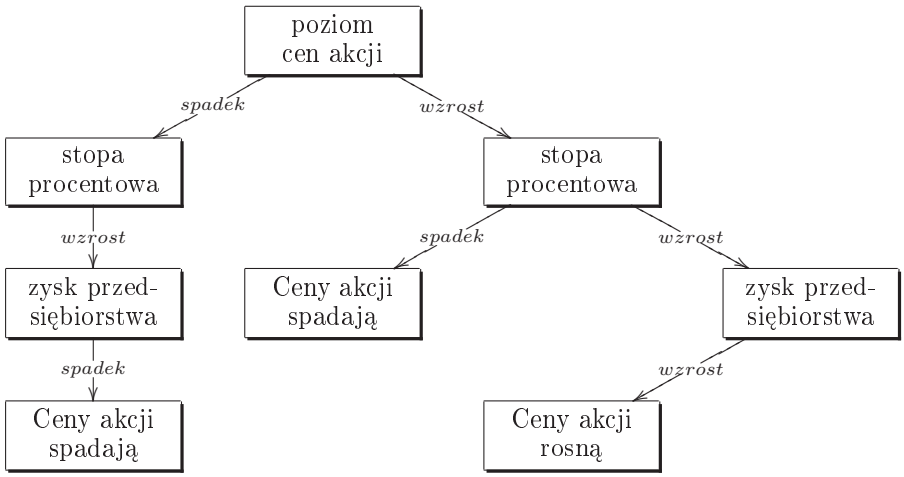
2 Drzewa klasyfikacyjne

Drzewem decyzyjnym (klasyfikacyjnym) określimy drzewo reprezentujące proces podziału zbioru obiektów na jednorodne klasy. W takim drzewie wewnętrzne węzły będą opisywać sposób dokonania podziału na jednorodne klasy (dokonywany w oparciu o wartości cech obiektów), a liście klasom, do których obiekty należą. Z kolei krawędzie drzewa reprezentują wartości cech, na podstawie których dokonano podziału. Przykład drzewa decyzyjnego przedstawia rysunek 1.

2.1 Proces tworzenia drzewa

Celem jest oczywiście zbudowanie drzewa jak najmniejszego (o minimalnej liczbie węzłów), po to by by otrzymane reguły klasyfikacji były jak najprostsze. Bardzo ogólna postać algorytmu składa się z następujących kroków:

[1] W literaturze częściej możemy spotkać określenia: drzewo decyzyjne. W statystyce często także: drzewa regresyjne, drzewa dyskryminacyjne



Rysunek 1: Drzewo decyzyjne - klasyfikacyjne

1. *Mając zbiór obiektów S, sprawdź, czy należą one do tej samej klasy. Jeśli tak, to zakończ pracę.*
2. *W przeciwnym przypadku rozważ wszystkie możliwe podziały zbioru S na podzbiory $S_1, S_2, ...S_n$ tak, aby były one jak najbardziej jednorodne.*
3. *Dokonaj oceny jakości każdego z tych podziałów zgodnie z przyjętym kryterium i wybierz najlepszy z nich.*
4. *Podziel zbiór S w wybrany sposób.*
5. *Wykonaj kroki 1-4 rekurencyjnie dla każdego z podzbiorów.*

Na podstawie drzewa klasyfikacyjnego można łatwo sformułować reguły przynależności obiektów do klas w odniesieniu do drzewa przedstawionego na rysunku 1:

- W przypadku spadku cen akcji: *"jeżeli stopa procentowa rośnie i zyski przedsiębiorstw spadają to ceny akcji spadają"*
- w przypadku wzrostu cen akcji: *"jeżeli stopa procentowa spada, lub jeśli stopa procentowa rośnie ale jednocześnie rosną zyski przedsiębiorstw rosną to ceny akcji rosną."*

2.2 Rodzaje drzew klasyfikacyjnych

Różnice dotyczą postaci funkcji oceniającej jakości podziału, sposobu klasyfikacji obiektów o brakujących wartościach cech, itd.

Najbardziej elementarny podział drzew decyzyjnych to podział na:

- drzewa binarne, w których z każdego wewnętrznego węzła wychodzą jedynie dwie krawędzie,

- drzewa niebinarne - gdzie z węzła mogą wychodzic więcej niż dwie krawędzie.

Tabela 1 [2] prezentuje znane algorytmy budowy drzew klasyfikacyjnych z podziałem na binarne i dowolne. Najpopularniejsze stosowane algorytmy to:

Tablica 1: Rodzaje algorytmów tworzenia drzew decyzyjnych

NAZWA	ROK	AUTORZY	RODZAJ DRZEWA
CLS	1996	Hunt,Marin, Stone	binarne
ACLS	1982	Paterson, Niblett	binarne
ID3	1983	Quinlan	dowolne
CART	1984	Brieman, Friedman	
		Olshen, Stone	binarne
ASSISTANT	1985	Kononenko	binarne
ID4	1986	Schlimmer, Fisdher	dowolne
PLS	1986	Rendell	dowolne
C4	1987	Quinlan	dowolne
GID 3	1988	Chengf, Fayyad,Irani	dowolne
ID5	1989	Utgoff	dowolne
LMDT	1991	Brodley, Utgoff	binarne, wielowymiarowe
CHAID	1993	SPSSInc.	dowolne
IND	1993	Bruntine, Caruana	dowolne
SADT	1993	Heat,Kasif,Salzberg	binarne, wielowymiarowe
SE-LEARN	1993	Rymonn	dowolne
OC1	1994	Murthy	binarne, wielowymiarowe

1. ID3 - cechujący się prostotą, ale wymagający kompletnych danych i nie pozwalający na szum w danych. Ponadto zakłada, że dane są danymi dyskretnymi, nie zaś ciągłymi.

2. C 4.5 - będący rozszerzeniem algorytmu ID3 i rozwiązujący większość problemów algorytmu ID3 (braki w danych, dane ciągłe, możliwość przycinania drzew gdy się zbytnio rozrastają (ang. pruning)).

3. CART (Classification and Regression Trees) - stosuje w budowie drzewa indeks Giniego, miarę entropii i regułę podziału na dwie części (twoing rule). Cechą charakterystyczną metody jest nadmierny rozrost drzewa i przycinanie (pruning) poszczególnych gałęzi w celu redukcji opisu liści (przy nieznacznym wzroście błędu klasyfikacji). Pozwala to na porównanie modelu rozbudowanego i modelu ze zredukowaną liczbą węzłów, czasami bowiem o jakości drzewa nie decyduje trafność predykcji, ale przydatność wygenerowanych reguł.

4. CHAID to algorytm AID (Automatic Interaction Detection) wykorzystujący test niezależności chi-kwadrat. Na każdym etapie podziału drzewa tworzy się tabelę kontyngencji, w której zestawia się zmienną objaśnianą (zależną) i objaśniającą. Jeśli zmienna objaśniana ma $d > 2$ kategorii, a objaśniająca $c > 2$ kategorii, to dąży się do redukcji tabeli kontyngencji o wymiarach $d \times c$ do bardziej istotnej (z punktu widzenia testu

[2] Źródło: Gatnar E.: "Symboliczne metody klasyfikacji danych", PWN, 1998, Polska

niezależności chi-kwadrat) o wymiarach $d \times j$, przez łączenie w dozwolony sposób kategorii zmiennej objaśniającej. Oryginalny CHAID pozwala budować modele dyskryminacyjne, czyli takie, których zmienna objaśniana jest zmienną nominalną.

2.3 Cel budowy drzew

Drzewo budujemy po to by potem móc klasyfikować nowe przypadki (przyszłe obserwacje), o których nie mamy informacji o przynależności klasowej. Budowane drzewo powinno być jak najmniejsze, większość algorytmów dodatkowo dokonuje porządkowania drzewa **(prunning)**, polegającego na usuwaniu tych jego fragmentów, które mają niewielkie znaczenie dla jakości rezultatów klasyfikacji.

2.4 Problemy ?

Każdy algorytm tworzący drzewa klasyfikacyjne musi zatem rozwiązać 3 problemy:

- jak wybrać jedną lub kilka cech, w oparciu o które nastąpi podział zbioru obiektów?
- kiedy zakończyć podział pozostałego podzbioru obiektów ?
- w jaki sposób przydzielić obiekty znajdujące się w liściu drzewa do pewnej klasy ?

3 Ważne aspekty budowy drzewa

Zasadniczym problemem jest wybór właściwego atrybutu do zbudowania całego testu. Najlepszy wybór to wybór takiego atrybutu, dla którego skrócimy ścieżkę w drzewie prowadzącą przez ten węzeł do liści wskazujących klasę decyzyjną.W tym celu, niezbędny jest wybór pewniej miary oceniającej, np. miarę przyrostu informacji (ang. *information gain*). Wykorzystywane jest przy tym zjawisko entropii. Jeśli S będzie zbiorem uczącym zawierającym n przykładów należących do jednej z k klas decyzyjnych oznaczonych przez $K_1, \ldots, K_k$, a n_i oznacza liczebność klasy K_i, wówczas **entropia** związana z klasyfikacją zbioru S jest zdefiniowana jako:

$$Ent(S) = -\sum_{i=1}^{k} p_i \lg_2 p_i$$

, gdzie p_i jest prawdopodobieństwem, że losowo wybrany przykład z S należy do klasy K_i, estymowanym jako $\frac{n_i}{n}$. Entropia podziału zbioru przykładów S ze względu na atrybut a jest zdefiniowana jako:

$$Ent(S|a) = \sum_{j=1}^{p} \frac{n_{S_j}}{n} Ent(S_j).$$

Można stwierdzić, że entropia $Ent(S|a)$ jest średnią ważoną dla entropii poszczególnych podzbiorów S_j. Im mniejsza wartość $Ent(S|a)$, tym większa jednorodność klasyfikacji dla przykładów podzielonych na podzbiory. **Przyrost**

informacji wynikający z zastosowania atrybutu a do zbudowania testu dzielącego zbiór przykładów uczących S jest zdefiniowany jako różnica:

$$Gain(S, a) = Ent(S) - Ent(S|a).$$

3.1 Przykład tworzenia drzewa

Załóżmy, że chcemy klasyfikować klientów sklepu elektronicznego pod względem tego czy kupią komputer czy nie. Elementy tego zbioru zestawiono w tabeli 2.

Tablica 2: Zbiór przykładów uczących opisujących grupę klientów sklepu elektronicznego

lp	Dochody	Student	Płeć	Kupuje komputer
1	średnie	tak	mężczyzna	tak
2	średnie	nie	kobieta	nie
3	wysokie	tak	kobieta	tak
4	niskie	tak	mężczyzna	nie
5	niskie	tak	kobieta	nie
6	średnie	tak	kobieta	tak
7	niskie	nie	kobieta	nie
8	średnie	nie	mężczyzna	nie

Wśród przykładów występuje binarna klasyfikacja. W związku z tym miara entropii dla zbioru S wyraża się wzorem:

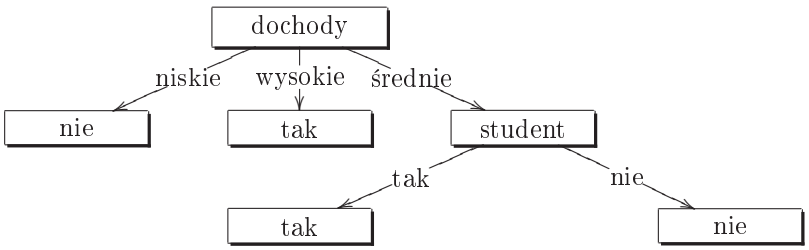
$$Ent(S) = -p_{Tak} \lg_2 p_{Tak} - p_{Nie} \lg_2 p_{Nie}$$

Zbiór 8 przykładów składa się z 3 przykładów decyzji Tak i 5 na decyzję Nie.
Odpowiednie prawdopodobieństwa są równe $p_{tak} = 3/8$ oraz $p_{nie} = 5/8$.
Wartość entropii związanej z binarną klasyfikacją rozważanego zbioru przykładów jest następująca:

$$Ent(S) = -(3/8) \lg_2(3/8) - (5/8) \lg_2(5/8) = 0.531 + 0.424 = 0.955$$

. Jeśli wybierzemy atrybut *dochody* do zbudowania korzenia drzewa, a ma on 3 wartości: $\{niskie, rednie, wysokie\}$.
Pierwszy podzbiór $S_{niskie} = \{4, 5, 7\}$ zawiera 3 przykłady, które należą do klasy decyzyjnej Nie.
Drugi podzbiór $S_{średnie} - \{1, 2, 6, 8\}$ zawiera po 2 przykłady z obu klas, podczas, gdy podzbiór $S_{wysokie} = \{3\}$ złożony jest z jednego przykłady z klasy Tak.
Wartość entropii warunkowej ze względu na ten atrybut jest następująca:
$Ent(S|dochody) = \frac{3}{8} * Ent(S_{niskie}) + \frac{4}{8} * Ent(S_{średnie}) + \frac{1}{8} * Ent(S_{wysokie}) =$
$\frac{(3}{8} * (-0 * \log_2 0 - 1 * \log_2 1) + \frac{4}{8}(-\frac{1}{2} * \log_2 \frac{1}{2} - \frac{1}{2} * \log_2 \frac{1}{2}) + \frac{1}{8} * (-0 * \log_2 0 - 1 * \log_2 1) =$
$0 + 0.5 + 0 = 0.5$
Przyrost informacji:
$GainInformation(S, dochody) = Ent(S) - Ent(S|dochody) = 0.955 - 0.5 =$
0.455.

Wartości miar przyrostu informacji wynikających z zastosowania pozostałych



Rysunek 2: Drzewo decyzyjne dla pojęcia "kupuję komputer".

atrybutów do budowy korzenia drzewa są następujące:
$Gain(S, student) = 0.348$ oraz $Gain(S, płeć) = 0.004$.
Podzbiory przykładów przypisane gałęziom odpowiadającym wartościom *niskie* oraz *wysokie* mają jednoznaczne przydziały do klas decyzyjnych, dlatego te gałęzie można zakończyć liśćmi etykietowanymi odpowiednio klasami *tak* i *nie*. W przypadku podzbiorów przykładów $S_{rednie} = \{1, 2, 6, 8\}$ należy rekurencyjnie wywołać algorytm. Z dwóch rozważanych atrybutów korzystniejszy przyrost informacji pozwala osiągnąć atrybut *student*, którego wartości jednoznacznie rozdzielają podzbiór przykładów na klasę *tak*(przykłady 1,6) oraz klasę *nie* (odpowiednio pozostałe przykłady 2,8) .

3.1.1 Problem z miarą *Information Gain*

Niestety miara przyrostu informacji (ang. *gain*) mając dwa atrybuty do wyboru, wybierze ten o większej liczbie wartości. Nie jest to pożądana właściwość, zwłaszcza w sytuacjach mocnego zróżnicowania liczności dziedzin atrybutów opisujących analizowane przykłady. Jeśli rozważymy skrajny przypadek, w którym pewien atrybut b, oznaczający np. datę urodzin, ma tyle różnych wartości, ile jest przykładów uczących, atrybut ten zostanie wybrany do zbudowania testu w węźle drzewa, gdyż maksymalizuje on wartość miary $Gain(S, b)$. W rezultacie każdy z podzbiorów S_i zawierać będzie pojedynczy przykład, co doprowadzi do stworzenia płaskiego i równocześnie bardzo szerokiego drzewa. Takie drzewo odwzorowuje dane uczące, lecz niestety jest mało czytelne dla użytkownika i równocześnie nie jest użyteczne do predykcji klasyfikacji tych przykładów, które nie są reprezentowane w zbiorze uczącym. Jeśli rozważymy test z wykorzystaniem atrybutu b, który oznaczał pytanie o datę urodzin, to zauważmy, ze takie pytanie pozostanie bez odpowiedzi dla nowych przykładów z inną wartością daty niż te, które wystąpiły w zbiorze uczącym.

3.1.2 Inne miary wyboru atrybutów do podziału drzewa

Wśród innych możliwych do zastosowania miar wyboru atrybutu do podziału drzewa są:

- *Split information* zwana **podziałem informacji** zaproponowana przez Quinlana, oceniająca podział zbioru przykładów ze względu na wartości z

dziedziny atrybutu a. Jest zdefiniowana w następujący sposób:

$$Split(S|a) = \sum_{j=1}^{r} \frac{|S_j|}{S} * \lg_2 \frac{S_j}{S},$$

gdzie S_j jest podzbiorem przykładów opisanych j-tą wartością atrybutu a, r jest liczbą różnych wartości w dziedzinie tego atrybutu.

- **ilorazem przyrostu informacji**(ang.*gain ratio*) zaproponowana również przez Quinlana jako miara do "normalizacji" przyrostu informacji i oceny jakości testu w węźle:

$$Gainratio(S|a) = \frac{Gain(S|a)}{Split(S|a)}.$$

Zasada wyboru atrybutu do stworzenia węzła w algorytmie indukcji drzew jest niezmieniona, tzn. zawsze wybierać będziemy ten atrybut, który pozwala maksymalizować wartość miary *Gain ratio*.

4 Binaryzacja drzew decyzyjnych

W przypadku, gdy mamy do czynienia z bardziej zróżnicowanymi danymi, (nie tylko jakościowymi) o małym zbiorze wartości, często modyfikuje się podstawowy schemat algorytmu, tak, aby generować binarne drzewa decyzyjne. **Binarne drzewo decyzyjne** charakteryzuje się tym, że z każdego jego wewnętrznego węzła wychodzą jedynie dwie krawędzie, czyli każdy zbiór przykładów związany z węzłem dzieli się na dwa rozłączne podzbiory. Taki rodzaj drzew ogranicza wystąpienie zjawiska fragmentacji danych, tj. stopniowego podziału zbioru przykładów na coraz mniejsze podzbiory, które mogą zawierać zbyt małą liczbę przykładów. Konstruowanie binarnych drzew decyzyjnych wiąże się z innymi sposobami tworzenia testów do umieszczenia w węźle drzew, tak, aby odpowiedzi na test były zawsze dwuwartościowe, np. prawda lub fałsz.

5 Postępowanie w przypadku brakujących wartości atrybutów

Rzeczywiste dane mogą zawierać nieznane (niezdefiniowane) wartości części atrybutów (ang. *unknown values of attributes*) dla niektórych obiektów. Sytuacje takie mogą wynikać z błędów podczas rejestracji danych, zagubienia zapisów bądź niedostępności pewnych informacji. Występowanie niezdefiniowanych wartości atrybutów wpływa zarówno na sam proces budowy drzewa, jak i na późniejsze użycie go do klasyfikowania nowych lub testowych obiektów. Część metod stosowana jest we wstępnym przetwarzaniu danych przed użyciem właściwego algorytmu indukcji. Wiele z nich jest ukierunkowanych na zastępowanie nieznanej wartości atrybutu dla określonego przykładu wartością z dziedziny tego atrybutu. Używa się najczęściej występującej wartości atrybutu, określonej na podstawie przykładów z pełnym opisem lub podzbioru tych przykładów należących do tej samej klasy decyzyjnej co analizowany przykład.

6 Budowa i analiza drzew klasyfikacyjnych w środowisku R

W pakiecie R metoda wyznaczania drzew decyzyjnych dostępna jest w wielu różnych funkcjach. Popularnie wykorzystywane są funkcje `tree(tree)`, `rpart(rpart)` oraz `cpart(party)`. Szczegółowe opisy pakietów są dostępne w lokalizacjach:

- `tree` `http://cran.r-project.org/web/packages/tree/tree.pdf`,
- `rpart` zaktualizowany 3 stycznia 2010 roku: `http://cran.r-project.org/web/packages/rpart/rpart.pdf`.

Gdybyśmy chcieli zbudować drzewo klasyfikacyjne dla dobrze znanego nam już zbioru `iris`, przy czym jako zbiór uczący wybrać chcielibysmy pierwsze 75 obserwacji formuła środowiska R do wykonania tego zadania będzie miała postać następującą:

```
> sub <-c(sample(1:150,75))
> library(rpart)
> fit<-rpart(Species~.,data=iris,subset=sub)
```

Efektem będzie tekstowo rozpisane drzewo z zagnieżdżeniami odpowiadającymi zagnieżdżeniom w drzewie (podwęzły).

```
> fit
n= 75

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

1) root 75 49 setosa (0.34666667 0.32000000 0.33333333)
  2) Petal.Length< 2.45 26  0 setosa (1.00000000 0.00000000 0.00000000) *
  3) Petal.Length>=2.45 49 24 virginica (0.00000000 0.48979592 0.51020408)
    6) Petal.Width< 1.55 22  0 versicolor (0.00000000 1.00000000 0.00000000) *
    7) Petal.Width>=1.55 27  2 virginica (0.00000000 0.07407407 0.92592593) *
```

Opis drzewa ma odpowiedni format:

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

gdzie: `node)` oznacza numer węzła, `split` - nazwę cechy, która dokonała podziału, `n` - liczbę elementów danego węzła, `loss` - liczbę błędnie klasyfikowanych elementów stosując regułę większościową. `yval` będzie odpowiadać wartości predykcji przynależności klasowej na podstawie reguły większościowej, `(yprob)` z kolei będzie przedstawiać wektor estymatorów prawdopodobieństw przynależności do danej klasy. Nazwy klas będą uporządkowane leksykograficznie. W opisie takim `*` oznacza element będący liściem w drzewie. Widzimy zatem, że drzewo tak utworzone ma 7 węzłów licząc z korzeniem drzewa. Widzimy także, że pierwszym atrybutem wybranym do budowy drzewa jest cecha `Petal.Length`, która w przypadku, gdy wartość `Petal.Length` jest mniejsza od 2.45 od razu prowadzi do klasyfikacji obiektu spełniającego ten warunek do klasy `setosa`. Takich elementów znaleziono 26. Jeśli zaś wartość `Petal.Length` jest większa

bądź równa wartości 2.45 wówczas będziemy musieli sprawdzić dodatkowy warunek dla cechy `Petal.Width`. Gdy teraz wartość tej cechy będzie mniejsza niż 1.55 obiekt zostanie przypisany do klasy `versicolor`, w przeciwnym przypadku do klasy `virginica`. Należy zwrócić uwagę na fakt, że takie drzewo zbudowano do 75 elementów ze zbioru `iris`. Dla innego drzewa, chociażby o większym rozmiarze powstałe drzewo może wyglądać zupełnie inaczej.

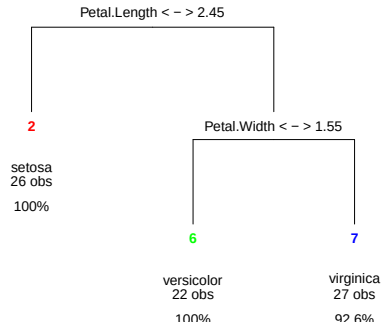
Jednak, jeśli takie przedstawienie drzewa decyzyjnego utworzonego dla zbioru `iris` jest dla nas nieczytelne możemy wykorzystać zasoby dodatkowych pakietów środowiska *R*, np `rattle`, który po zainstalowaniu musi być oczywiście załadowany do środowiska *R*. Dokonujemy tego wywołując jedną komendę *R*:

```
> library(rattle)
```

Teraz wywołując już komendę pakietu `rattle` o nazwie `drawTreeNodes` a konkretnie:

```
drawTreeNodes(fit, col = NULL, nodeinfo = FALSE,decimals = 2,
print.levels = TRUE, new = TRUE)
```

otrzymujemy w efekcie drzewo takie jak przedstawia to rysunek 3.



Rysunek 3: Drzewo decyzyjne

Możemy także posilić się standardowymi funkcjami środowiska *R*, które dostarczają bardzo szczegółowych wyników. Co ciekawe wyrysowanie wynikowego drzewa jest możliwe przez wywołanie kolejno po sobie dwóch komend *R*: `plot(fit)` oraz `text(fit)`. Efektem tego będzie otrzymane drzewo decyzyjne w tzw. formacie tekstowym. Szczegółowych wyników zaś dostarcza funkcja `summary`, której wywołanie i wyniki tego wywołania możemy zauważyć poniżej.

```
> summary(fit)
Call:
rpart(formula = Species ~ ., data = iris, subset = sub)
```

```
  n= 75

        CP nsplit  rel error     xerror       xstd
1 0.5102041      0 1.00000000 1.18367347 0.07399660
2 0.4489796      1 0.48979592 0.57142857 0.08548723
3 0.0100000      2 0.04081633 0.06122449 0.03463380

Node number 1: 75 observations,    complexity param=0.5102041
  predicted class=setosa      expected loss=0.6533333
    class counts:    26    24    25
   probabilities: 0.347 0.320 0.333
  left son=2 (26 obs) right son=3 (49 obs)
  Primary splits:
      Petal.Length < 2.45 to the left,  improve=25.48354, (0 missing)
      Petal.Width  < 0.8  to the left,  improve=25.48354, (0 missing)
      Sepal.Length < 5.55 to the left,  improve=15.29298, (0 missing)
      Sepal.Width  < 3.25 to the right, improve=12.71602, (0 missing)
  Surrogate splits:
      Petal.Width  < 0.8  to the left,  agree=1.000, adj=1.000, (0 split)
      Sepal.Length < 5.45 to the left,  agree=0.880, adj=0.654, (0 split)
      Sepal.Width  < 3.25 to the right, agree=0.867, adj=0.615, (0 split)

Node number 2: 26 observations
  predicted class=setosa      expected loss=0
    class counts:    26     0     0
   probabilities: 1.000 0.000 0.000

Node number 3: 49 observations,    complexity param=0.4489796
  predicted class=virginica   expected loss=0.4897959
    class counts:     0    24    25
   probabilities: 0.000 0.490 0.510
  left son=6 (22 obs) right son=7 (27 obs)
  Primary splits:
      Petal.Width  < 1.55 to the left,  improve=20.786090, (0 missing)
      Petal.Length < 4.85 to the left,  improve=17.143130, (0 missing)
      Sepal.Length < 6.25 to the left,  improve= 6.369796, (0 missing)
      Sepal.Width  < 2.95 to the left,  improve= 1.320830, (0 missing)
  Surrogate splits:
      Petal.Length < 4.75 to the left,  agree=0.939, adj=0.864, (0 split)
      Sepal.Length < 5.75 to the left,  agree=0.755, adj=0.455, (0 split)
      Sepal.Width  < 2.45 to the left,  agree=0.653, adj=0.227, (0 split)

Node number 6: 22 observations
  predicted class=versicolor  expected loss=0
    class counts:     0    22     0
   probabilities: 0.000 1.000 0.000

Node number 7: 27 observations
  predicted class=virginica   expected loss=0.07407407
    class counts:     0     2    25
   probabilities: 0.000 0.074 0.926

>
```

Graficzne przedstawienie reguły klasyfikacyjnej zadanej przez drzewo możliwe jest dzięki instrukcji R postaci `rpart(y~x,data="",cp=0.03,minisplit=5)`, która dla naszego zbioru `iris` może wyglądać następująco:

```
rpart(Species~.,data=iris,cp=0.03)
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) *
    7) Petal.Width>=1.75 46   1 virginica (0.00000000 0.02173913 0.97826087) *
>
```

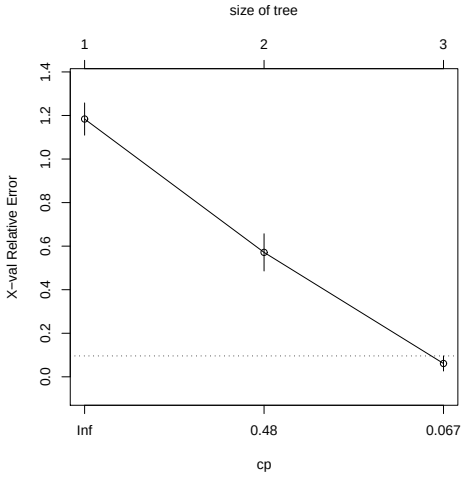
Zagadnieniem niezwykle istotnym jest określenie kryterium budowy drzewa optymalnego. Należy sobie zadać pytanie, co rozumiemy przez drzewo optymalne, czy jest to drzewo o najmniejszej liczbie węzłów, czy może drzewo o najmniejszej wysokości, czy jeszcze inne warunki będą określać optymalność drzewa ? Proponujemy, wykorzystanie informacji o błędach krosswalidacyjnych. Przy użyciu funkcji `printcp` możemy otrzymać informacje o wielkościach poddrzew optymalnych w zależności od wartości *cp* (patrz na kod poniżej).

```
> printcp(fit)

Classification tree:
rpart(formula = Species ~ ., data = iris, subset = sub)

Variables actually used in tree construction:
[1] Petal.Length Petal.Width

Root node error: 49/75 = 0.65333

n= 75

       CP nsplit rel error   xerror     xstd
1 0.51020      0  1.000000 1.183673 0.073997
2 0.44898      1  0.489796 0.571429 0.085487
3 0.01000      2  0.040816 0.061224 0.034634
> plotcp(fit)
>
```

Efektem będzie właśnie wykres tych błędow krosswalidacyjnych (rysunek 4)

Funkcja `printcp` zwraca wartość `xerror`, która jest ilorazem SSE_{cv} dla danego drzewa i SSE dla korzenia. Zwraca też błąd standardowy *std* (`xstd`). Będziemy ostatecznie wybierać jedną z dwóch opcji:

1. drzewo z najmniejszą wartością `xerror` ($xerror_{min}$),
2. drzewo o najmniejszej liczbie podziałów, gdzie $xerror < xerror_{min} + xstd_{min}$.



Rysunek 4: Wykres błędów krosswalidacyjnych

Minimalna wartość `xerror` w naszym przypadku to 0.061224, więc drzewo o minimalnej liczbie liści musi mieć `xerror` mniejszy niż $\texttt{xerror}_{min} + xstd_{min} = 0.061 + 0.034 = 0.095$. Czyli szukamy drzewa, które ma wartość *xerror* mniejszą niż 0.095. Będzie to opcja z 1 lub 2 podziałami.

7 Bibliografia

Opracowanie przygotowano w oparciu o prace:

1. J. Koronacki, J. Ćwik: *Statystyczne systemy uczące się*, wydanie drugie, Exit, Warsaw, 2008, rozdział 4.

2. J. Ćwik, J. Mielniczuk: *Statystyczne systemy uczące się - ćwiczenia w oparciu o pakiet R*, Oficyna Wydawnicza PW, Warszawa, 2009.

3. Biecek P.: *Na przełaj przez Data Mining*, `http://www.biecek.pl/R/naPrzelajPrzezDM.pdf`

4. Koronacki J. and Mielniczuk J., *Statystyka dla studentów kierunków technicznych i przyrodniczych*. Wydawnictwa Naukowo-Techniczne, Warszawa, Polska, 2006.