Uniwersytet Warszawski
Wydział Matematyki, Informatyki i Mechaniki

Dominika Nowicka
Nr albumu: 278552

Algorytmy przycinania drzew klasyfikacyjnych wraz z przykładami

Praca licencjacka
na kierunku MATEMATYKA W RAMACH MIĘDZYWYDZIAŁOWYCH INDYWIDUALNYCH STUDIÓW MATEMATYCZNO - PRZYRODNICZYCH w zakresie ZASTOSOWAŃ MATEMATYKI

Praca wykonana pod kierunkiem
dra inż. Przemysława Biecka
Instytut Matematyki Stosowanej i Mechaniki

Sierpień 2011

Oświadczenie kierującego pracą

Potwierdzam, że niniejsza praca została przygotowana pod moim kierunkiem i kwalifikuje się do przedstawienia jej w postępowaniu o nadanie tytułu zawodowego.

Data Podpis kierującego pracą

Oświadczenie autora (autorów) pracy

Świadom odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam również, że przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego w wyższej uczelni.

Oświadczam ponadto, że niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną.

Data Podpis autora (autorów) pracy

Streszczenie

Głównym tematem niniejszej pracy jest przycinanie drzew decyzyjnych. Praca składa się z trzech głównych rozdziałów. Każdy z nich podchodzi do zagadnienia z innej strony. Pierwszy jest poświęcony teorii, a więc opisuje algorytmy stosowane do przycinania drzew. W rozdziale drugim znaleźć można wiele przydatnych funkcji środowiska R dotyczących drzew klasyfikacyjnych oraz przykładów ich użycia. Trzeci rozdział stanowi połączenie obydwu tych podejść. Drzewa klasyfikacyjne zostały tam zastosowane do rzeczywisych danych pochodzących z badania 642 pacjentów pięciu europejskich zakładów psychiatrycznych.

Słowa kluczowe

drzewo klasyfikacyjne, próba ucząca, próba trenująca, przeuczenie, przycinanie, R, podział, kroswalidacja, parametr złożoności

Dziedzina pracy (kody wg programu Socrates-Erasmus)

11.2 Statystyka

Klasyfikacja tematyczna

62H30 Classification and discrimination; cluster analysis
92C50 Medical applications

Tytuł pracy w języku angielskim

An algorithms of pruning classification trees with examples.

Spis treści

Wprowadzenie

W dzisiejszym świecie analizowanie ogromnych zbiorów danych jest już standardem. Spotykamy je w wielu dziedzinach nauki i przemysłu. Mogą to być wyniki badania psychologicznego czy socjologicznego, wszelkie sondaże, dane kliniczne czy dane sprzedażowe. W związku z tym rozwijanych jest wiele metod służących do pracy na dużych zbiorach danych. Jedną z takich metod jest użycie drzew klasyfikacyjnych, których dotyczy niniejsza praca. Drzewa klasyfikacyjne pojawiły się po raz pierwszy w literaturze w latach sześćdziesiątych w kontekście badań socjologicznych. Wśród statystyków stały się popularne za sprawą książki Breimana i in. *Classification and Regression Trees* z roku 1984.

Głównym problemem omawianym w tej pracy jest *overfitting*, czyli przeuczenie drzewa klasyfikacyjnego. Przedstawiam tu dwa algorytmy służące do poprawy jakości drzewa i pokazuję, jak z nich korzystać w środowisku programistycznym R. Całość składa się z trzech rozdziałów. Pierwszy z nich poświęcony jest wprowadzeniu w tematykę drzew klasyfikacyjnych oraz teoretycznemu aspektowi omawianego zagadnienia. Wyjaśniam tam pokrótce, czym jest i jak powstaje drzewo oraz opisuję algorytmy używane do przycinania drzew. W drugim rozdziale przedstawione są natomiast pakiety i funkcje programu R, które pozwalają na pracę z drzewami klasyfikacyjnymi. Przedstawiłam tam zarówno sposoby wywoływania funkcji, jak i przykładowe efekty ich działania, ilustrując je fragmentami kodu z R oraz wykresami. Ostatni rozdział miał na celu pokazanie, jak drzewa decyzyjne działają na rzeczywistych danych. Przedstawiłam tam konkretny problem klasyfikacyjny. Dysponując danymi dotyczącymi pacjentów pięciu europejskich szpitali psychiatrycznych, starałam się udzielić odpowiedzi na pytanie, co wpływa na ocenę jakości życia dokonywaną przez te osoby.

Rozdział 1

Podstawy teoretyczne przycinania drzew decyzyjnych

W tym rozdziale omówię teoretyczne aspekty przycinania drzew klasyfikacyjnych. Przedstawię dwa algorytmy wykorzystywane w tym celu: prosty algorytm oparty na ułamku błędnych klasyfikacji i bardziej zaawansowany algorytm oparty na tzw. *kryterium kosztu-złożoności*. Zanim przejdę jednak do ich opisu, chcę przedstawić zarys procedury konstruowania drzewa i klasyfikacji z jego użyciem[1] , gdyż jest to niezbędne do zrozumienia istoty samej procedury przycinania drzewa.

1.1. Podstawowe pojęcia

Sekcja ta ma na celu przedstawienie od strony matematycznej zagadnień, o których będzie mowa w dalszej części pracy.

Drzewa klasyfikacyjne najprościej zdefiniować używając terminologii teorii grafów, stąd też zacznę od podstawowych definicji z tej dziedziny.

Definicja 1.1.1 Graf *to uporządkowana para G=(W,K), gdzie W jest niepustym zbiorem, którego elementy nazywamy* wierzchołkami*, a K jest rodziną dwuelementowych podzbiorów zbioru wierzchołków W zwanych* krawędziami*: $K \subseteq \{\{a,b\} : a,b \in W, a \neq b\}$.*

Definicja 1.1.2 *Graf* skierowany *to uporządkowana para G=(W,K), gdzie W jest niepustym zbiorem, którego elementy nazywamy* wierzchołkami*, a K jest zbiorem uporządkowanych par elementów ze zbioru wierzchołków* (krawędzi skierowanych)*: $K \subseteq W \times W$.*

Definicja 1.1.3 Drogą od a_1 do a_n *nazywamy taki uporządkowany podzbiór $H \subseteq G$,*
$H = (\{a_1, a_2, ..., a_n\}, \{k_1, k_2, ..., k_{n-1}\})$ dla pewnego n naturalnego, że $\forall_{i=1,2,...,n}\ k_i = \{a_i, a_{i+1}\}$. W takiej drodze a_1 nazywamy początkiem *drogi, a_n zaś jej* końcem*. Jeśli wszystkie k_i są parami różne, to drogę H nazywamy* prostą*.*

Definicja 1.1.4 *Graf nazywamy* spójnym*, gdy $\forall_{a,b \in W}\ \exists$ droga od a do b.*

Definicja 1.1.5 Cyklem *nazywamy taką drogę, w której $a_1 = a_n$.*

Definicja 1.1.6 *Graf* cykliczny *to graf, który zawiera co najmniej jeden cykl. Graf, który nie posiada cykli, nazywamy* acyklicznym*.*

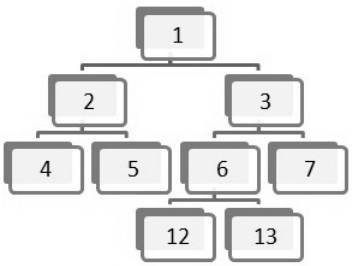
[1] Można na ten temat więcej przeczytać w literaturze, a w szczególności w pracy licencjackiej komplementarnej do niniejszej autorstwa Marty Tyce.

Przy użyciu powyższych pojęć łatwo już powiedzieć, czym jest drzewo.

Definicja 1.1.7 Drzewem *nazywamy graf, który jest acykliczny i spójny.*

Definicja 1.1.8 Drzewo klasyfikacyjne (decyzyjne) *to drzewo skierowane z wyróżnionym wierzchołkiem początkowym* - korzeniem*, będące* klasyfikatorem *(patrz:1.2.1).*

Elementy drzewa decyzyjnego najprościej jest przedstawić za pomocą rysunku.



Rys. 1. Przykład drzewa binarnego - oznaczmy je przez T.

Na powyższym rysunku:

- węzeł 1 jest *korzeniem* drzewa,
- węzły 2 i 3 są *dziećmi* węzła 1,
- węzeł 1 jest wobec tego *rodzicem* węzłów 2 i 3,
- węzły 4, 5, 7, 12 i 13 są *liśćmi* drzewa,
- krawędzie, którymi połączone są wszystkie węzły nazywamy *gałęziami.*

W praktyce zwykle używa się *drzew binarnych* (takich, w których każdy węzeł-rodzic ma dwoje dzieci), toteż takie drzewo znalazło się w przykładzie. Jak widać, niewątpliwą zaletą drzew binarnych jest łatwość numerowania kolejnych węzłów:

- Numer węzła = (numer węzła-rodzica) * 2 — dla lewego węzła-dziecka,
- Numer węzła = (numer węzła-rodzica) * 2 + 1 — dla prawego węzła-dziecka.

1.2. Krótkie omówienie budowy i działania drzewa decyzyjnego

Postaram się tutaj przedstawić opisowo sposób działania drzewa decyzyjnego oraz proces jego powstawania. Przydatne będą kolejne definicje...

Definicja 1.2.1 Klasyfikatorem *nazywamy odwzorowanie* $h : \mathbf{X} \mapsto Y$*, gdzie* $\mathbf{X}$ *jest ustalonym zbiorem wektorów* atrybutów*, a* Y *jest zbiorem* klas*.*

W pracy tej często nazywam drzewo decyzyjne klasyfikatorem, a więc zgodnie z powyższą definicją pewnym odzorowaniem. Wcześniej jednak definiowałam drzewo jako graf. Ściśle rzecz ujmując, drzewo decyzyjne tak zdefiniowane jest jedynie graficzną reprezentacją klasyfikatora, ale w praktyce również sam klasyfikator przyjęło się nazywać drzewem.

Definicja 1.2.2 Próba ucząca (trenująca) *to zbiór danych* $E = \{\mathbf{x_i}, y_i\}_{i \in I}$ *dla pewnego* I *— zbioru indeksów, gdzie parę* $(\mathbf{x_i}, y_i)$ *nazywamy* obserwacją, $\mathbf{x_i}$ *to wektor zmiennych opisujących i-tą obserwację* - atrybutów, *a* y_i *to pewna* klasa *ze zbioru* Y.

Innymi słowy próba ucząca to zbiór danych, składający się z pewnej liczby obserwacji opisanych ustaloną liczbą zmiennych, z których przynajmniej jedna jest *zmienną kategoryczną*, a więc przyjmuje skończoną liczbę wartości (klas). Do klasyfikowania tej zmiennej ma służyć budowany klasyfikator. Na próbę uczącą wygodnie jest patrzeć jak na tabelę, w której wierszach mamy obserwacje, a w kolumnach atrybuty. W próbie uczącej znane są oczywiście wszystkie wartości $\mathbf{x_i}$, y_i.

1.2.1. Trenowanie drzewa

Zacznijmy od definicji...

Definicja 1.2.3 Trenowanie drzewa *to proces tworzenia klasyfikatora, jakim jest drzewo decyzyjne, przy użyciu próby uczącej.*

Załóżmy, że nasze drzewo ma dawać informację o poziomie pewnej cechy K (K jest zmienną binarną: K=niski lub K=wysoki) oraz dysponujemy próbą trenującą ($\mathbf{X} = (A, B, C, D)$, Y= K).

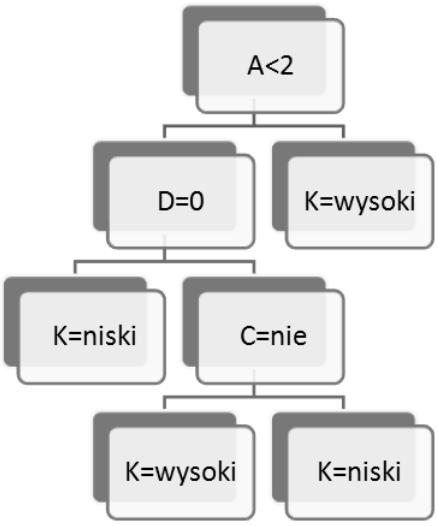
Zanim przystąpimy do budowy drzewa, powinniśmy wybrać, jaką miarą różorodności będziemy się posługiwać. Skoro mamy łączyć w grupy "podobne" obserwacje, a oddzielać od siebie "różne", musimy ustalić, jak mierzyć to zróżnicowanie. Celem jest, aby wewnątrz węzła-dziecka zróżnicowanie było jak najmniejsze, a między węzłami-dziećmi jak największe. Istnieje wiele miar różnorodności, o których nie będę tu więcej pisać. [2]

Proces wzrostu drzewa, jak łatwo się domyślić, zaczyna się od korzenia. W tym pierwszym węźle znajduje się cała próba ucząca. Tam też odbywa się jej podział na 2 części tak, by zgodnie z obraną miarą różnorodności węzły 2 i 3 jak najbardziej się od siebie różniły. Następnie tak samo dzielimy wszystkie kolejne węzły. Jeśli nie narzucimy żadnych ograniczeń na rozbudowę drzewa, proces ten będzie trwał aż do uzyskania drzewa z liśćmi, w których znajdują się obserwacje z jednej tylko klasy (np. tej o niskim poziomie K) lub obserwacje o identycznych wartościach zmiennych A, B, C i D lecz o różnych poziomach K (drzewo pełne).

1.2.2. Jak klasyfikuje drzewo?

Załóżmy, że dysponujemy już pewnym drzewem decyzyjnym. Drzewo to powstało na podstawie zbioru uczącego takiego jak wyżej (opisanego zmiennymi A, B, C, D, K, gdzie klasyfikowaną zmienną jest zmienna K).

[2] Odsyłam do [KC] i do pracy licencjackiej komplemenatrnej do niniejszej autorstwa Marty Tyce.



Rys. 2. Drzewo klasyfikacyjne - przykład reguł klasyfikacji.

Nowa obserwacja jest przypisana do jednej z dwóch klas (niskie K, wysokie K) w następujący sposób:

1. węzeł 1: jeśli $A < 2$, to obserwacja "idzie" na lewo, jeśli nie - na prawo (i tutaj obserwacja kończy swoją "trasę", to znaczy, że ma wysoki poziom K),

2. węzeł 2: jeśli obserwacja ma $D = 0$, "idzie" na lewo (i zostaje zaklasyfikowana do grupy z niskim poziomem K), jeśli $D \neq 0$ - na prawo,

3. węzeł 5: jeśli obserwacja ma $C = nie$, to (zgodnie z predykcją drzewa) ma wysoki poziom K, w przeciwnym przypadku - niski.

1.3. Przycinanie drzew decyzyjnych

Należy pamiętać, że ogromny wpływ na wyjściową postać klasyfikatora, jakim jest drzewo decyzyjne, ma próba ucząca. Nadmierne do niej dopasowanie, zwane *przetrenowaniem drzewa*, powoduje nie najlepsze zdolności klasyfikacyjne takiego drzewa, a więc duże błędy w klasyfikacjach przez nie dokonywanych. Zatem słabo klasyfikuje *drzewo pełne*, to znaczy takie, które w każdym liściu ma elementy z jednej tylko klasy, bądź też elementy z różnych klas, ale o takim samym wektorze wartości. Aby poprawić jakość klasyfikatora możemy narzucić ograniczenia na rozbudowę drzewa poprzez określenie minimalnej liczby elementów w liściu lub maksymalnej wysokości drzewa. Przede wszystkim jednak stosujemy procedurę zwaną *przycinaniem drzewa*, polegającą na usuwaniu końcowych gałęzi drzewa, a potem ich poprzedników aż do uzyskania największej możliwej zdolności klasyfikacyjnej drzewa.

Przydatne będą następujące definicje.

Definicja 1.3.1 Przeuczeniem (przetrenowaniem) drzewa *nazywamy zbytnie dopasowanie drzewa do próby uczącej powodujące duże błędy w klasyfikacji popełniane przez to drzewo.*

Definicja 1.3.2 Przycinaniem drzewa *nazywamy proces usuwania* (odcinania) *końcowych gałęzi drzewa, a potem ich poprzedników prowadzący do uzyskania największej możliwej zdolności klasyfikacyjnej drzewa.*

Definicja 1.3.3 Poddrzewem *drzewa T nazywamy takie drzewo, którego wszystkie węzły są też węzłami drzewa T.* Poddrzewo zakorzenione *drzewa T to takie jego poddrzewo, które ma z drzewem T wspólny korzeń.*

Definicja 1.3.4 Drzewem pełnym *nazywać będziemy drzewo klasyfikacyjne, które powstało bez żadnych ograniczeń na rozbudowę, a więc mające w liściach pojedyncze obserwacje bądź obserwacje o takim samym wektorze atrybutów, a różniące się tylko klasą, do której należą.*

Definicja 1.3.5 Próba testowa *to zbiór danych* $E = \{\mathbf{x_i}, y_i\}_{i \in I}$ *dla pewnego* I *— zbioru indeksów, gdzie* $\mathbf{x_i}$ *to wektor atrybutów z tego samego zbioru* $\mathbf{X}$*, co w przypadku próby uczącej,* y_i *to klasa z tego samego, co w przypadku próby uczącej, zbioru* Y*, ale obserwacje* $(\mathbf{x_i}, y_i)$ *różnią się od tych z próby uczącej.*

Próba testowa to dane, które klasyfikujemy przy użyciu stworzonego wcześniej klasyfikatora w celu określenia jego zdolności klasyfikacyjnych. Również w przypadku próby testowej znamy wszystkie wartości $\mathbf{x_i}$, y_i, co jest warunkiem koniecznym do porównania rzeczywistej i wskazywanej przez drzewo klasy.

1.3.1. Algorytm oparty na ułamku błędnych klasyfikacji

Najbardziej podstawowym algorytmem przycinania drzew jest ten oparty na ułamku błędnych klasyfikacji, a więc stosunku źle zaklasyfikowanych obserwacji z próby testowej do wszystkich obserwacji z tej próby (bądź też liczbie błędnych klasyfikacji, bo wyniki są oczywiście przy obydwu podejściach takie same).

Definicja 1.3.6 Rozmiarem drzewa *nazywamy liczbę jego liści. Dla drzewa T będziemy go oznaczać przez* $| T |$.

Za miarę zdolności klasyfikacyjnych drzewa na danej próbie przyjmiemy liczbę błędnych klasyfikacji. Innymi słowy: A jest lepszym klasyfikatorem niż B, jeśli liczba błędów popełnionych przez A jest mniejsza niż liczba błędów popełnionych przez B.

Załóżmy, że dysponujemy pewnym drzewem klasyfikacyjnym. Może to być drzewo pełne, bądź też drzewo powstałe z narzuceniem warunków stopu. Nazwijmy je drzewem startowym i oznaczmy przez T_0. Będziemy chcieli tak zmniejszyć to drzewo, aby popełniało jak najmniej błędów podczas klasyfikacji. W tym celu będziemy chcieli sprawdzić wszystkie poddrzewa drzewa T_0. Będziemy sprawdzać kolejno drzewa o coraz mniejszej liczbie liści, tworząc ciąg drzew T_1, T_2, ..., $T_{|T_0|}$, z których każde popełnia najmniej błędów wśród drzew o takim samym rozmiarze.

Poniżej przedstawiam zapis algorytmu w pseudokodzie.

`trenuj` T_0`,`
`n=`$| T_0 |$ `err=wektor błędów popełnianych przez kolejne drzewa z powstającego ciągu` T_k
`T=wektor kolejnych drzew` T_k
`er=moc zbioru testowego`
% więcej błędów niż liczba obserwacji drzewo nie popełni
`drz=wektor składający się z wektorów poddrzew` T_0
% drz[k] będzie wektorem poddrzew T_0 o $n-k$ liściach, wektor ten będzie miał długość $\binom{n}{k}$
`j=` T_0

```
for (k in 0:n-1) {
    drz[k]=wektor poddrzew T_0 o n-k liściach
    for ( i in drz[k]) {
       klasyfikuj zbiór testowy przy pomocy i
       eri= liczba obserwacji, które drzewo i źle zaklasyfikowało
       if (eri<er) {
          err[k]=eri
          T[k]=i
       }
    }
}
m= max_{1≤k≤n} err[k]
T=T[m]
```

W ten sposób uzyskaliśmy drzewo, które na zadanej próbie uczącej robiło najmniej błędów i to właśnie drzewo będziemy uważać za optymalne.

Należy zauważyć, że w powyższym algorytmie powstała rodzina drzew nie musi być rodziną zagnieżdżoną, to znaczy, że kolejne drzewa nie muszą być poddrzewami poprzednich. Może to utrudniać płynne przechodzenie od większych drzew do mniejszych i porównywanie ich między sobą.

1.3.2. Algorytm oparty na kryterium kosztu-złożoności

Opisana niżej metoda pozbawiona jest wady, jaką ma poprzedni algorytm, a zatem pozwala ona na konstrukcję zagnieżdżonej rodziny poddrzew (zstępującego ich ciągu).

Niech Q(T) będzie miarą niedoskonałości drzewa T — u nas będzie to ułamek błędnych klasyfikacji. Dla każdego α (nieujemnego współczynnika złożoności) szukamy takiego drzewa T zakorzenionego w drzewie pełnym T_0, że wartość minimalną osiąga funkcja kosztu–złożoności wyrażona wzorem:

$$S_\alpha(T) = Q(T) + \alpha * \mid T \mid . \tag{1.1}$$

Należy tu zauważyć dwie rzeczy:

1. dla ustalonego α odcinanie kolejnych gałęzi drzewa powoduje wzrost Q(T) i zmniejszanie $\alpha * \mid T \mid$,
2. im większy współczynnik α, tym mniejsze drzewa wybierane są jako optymalne, dla $\alpha = 0$ wybierane jest drzewo pełne, a od pewnego α wybierany jest już sam korzeń.

Przycinanie drzewa pełnego T_0 przebiega dwuetapowo...

1. Pierwszy etap polega na skonstruowaniu takiego jak wyżej ciągu poddrzew minimalizujących funkcję kosztu-złożoności (przy konstrukcji j-te drzewo buduje się na podstawie dowolnie ustalonej wartości $\alpha'_j \in [\alpha_j, \alpha_{j+1})$). Koniecznie jest zauważenie, że dla danego α'_j drzewo optymalne nie musi być wyznaczone jednoznacznie. Wybieramy najmniejsze z takich drzew i oznaczamy je jako T_j. Wszystkie możliwe wartości α można podzielić na rozłączne przedziały, wewnątrz których dla wszystkich α wybierane jest to samo drzewo. I tak:

- $T(\alpha) = T_0 \; dla \; \alpha < \alpha_1$,
- $T(\alpha) = T_k \; dla \; \alpha_k \leqslant \alpha \leqslant \alpha_{k+1}, 1 \leqslant k \leqslant K$,
- $T(\alpha) = T_K \; dla \; \alpha \geqslant \alpha_K$,

gdzie K jest najmniejszym indeksem, dla którego wybrany został sam korzeń.

2. Etap drugi ma na celu wybranie z ciągu $T_1, T_2, ..., T_K$ drzewa T_{j_0} o najmniejszym ułamku błędów klasyfikacji dla próby testowej.

   W przypadku, gdy nie dysponujemy próbą testową, wybór najlepszego drzewa odbywa się na podstawie *kroswalidacji*:

   (a) Dzielimy próbę uczącą na n równych części.

   (b) Tworzymy i-tą podpróbę poprzez usunięcie z próby uczącej i-tej części.

   (c) Na podstawie każdej takiej podpróby budujemy i-te drzewo minimalizujące kryterium kosztu-złożoności dla ustalonego α'_j.

   (d) Testujemy je następnie na i-tej części wyjściowej próby.

   (e) Sumujemy liczby błędów klasyfikacji wszystkich n powstałych drzew $(T_{j_1}, T_{j_2}, ..., T_{j_n})$ i uzyskujemy w ten sposób kroswalidacyjne oszacowanie ułamka błędnych klasyfikacji $(Q_s(T_j))$ drzewa $T_j = T(\alpha'_j)$.

$$Q_s(T_j) = \sum_{i=1}^{n} Q(T_{j_i}) \tag{1.2}$$

   (f) Ostatecznie wybieramy z oryginalnego ciągu drzew T_j drzewo T_{j_0}, które ma najmniejszy sumaryczny ułamek błędów. Jednak uwzględniając losowość kroswalidacji, Breiman sugerował, by jako optymalne drzewo wybierać najmniejsze z drzew leżących w odległości jednego odchylenia standardowego sumarycznego ułamka błędnych klasyfikacji tego drzewa, które minimalizuje błąd klasyfikacji (*reguła 1SE)*. Estymatorem odchylenia standardowego jest tutaj:

$$\left(\frac{Q_s(T_{j_0})(1 - Q_s(T_{j_0}))}{n} \right)^{1/2}. \tag{1.3}$$

Rozdział 2

Polecenia w pakiecie R

W tej części pracy przedstawię funkcje pakietu R, które umożliwiają pracę z drzewami decyzyjnymi. R posiada kilka wbudowanych pakietów do tego właśnie służących. Generalnie pomocne są pakiety:

- rpart,
- tree,
- party,
- maptree.

2.1. Podstawowe funkcje w R i ich parametry

Prostym sposobem uzyskania informacji na temat funkcji związanych z danym pakietem (dla przykładu niech to będzie pakiet *rpart*) bezpośrednio z wiersza poleceń jest komenda:

```
??rpart.
```

Jeśli natomiast chcemy uzyskać informację odnośnie konkretnej funkcji (funkcja o nazwie *rpart* również istnieje, więc możemy pozostać przy tym przykładzie), można to zrobić poleceniem:

```
?rpart.
```

Do stworzenia drzewa w programie R służyć mogą funkcje takie jak: *tree()* z pakietu tree, *ctree()* z pakietu party oraz *rpart()* z pakietu rpart. Przedstawię teraz te ich argumenty, które są istotne dla kontroli rozrostu drzewa. Argumenty tych komedn są do siebie podobne, ale nie są identyczne, dlatego też dla porządku omówię je oddzielnie. Szczegółowy opis pozostałych znaleźć można w ogólnodostępnej dokumentacji pakietu R.

2.1.1. Podstawowe funkcje budujące drzewo

Przyjrzyjmy się teraz funkcji *ctree* z pakietu party.

```
ctree(formula, data, subset = NULL, weights = NULL,
      controls = ctree_control(), xtrafo = ptrafo, ytrafo = ptrafo,
      scores = NULL)
```

formula	symboliczna specyfikacja modelu
data	tabela z danymi do modelu (próbą uczącą)
controls	obiekt klasy TreeControl, który opisuję niżej

```
ctree_control(mincriterion = 0.95, minsplit = 20, minbucket = 7,
              stump = FALSE, maxdepth = 0)
```

mincriterion	wartość statystyki testowej lub 1-p-wartość, która musi być osiągnięta, aby wykonać podział
minsplit	minimalna liczba obserwacji w węźle, aby dokonać jego podziału
minbucket	minimalna liczba obserwcaji w liściu
maxdepth	maksymalna wysokość drzewa, domyślna wartość *maxdepth=0* oznacza brak ograniczeń

W pakiecie rpart analogiczną rolę pełni funkcja:

```
rpart(formula, data, na.action = na.rpart, method, control, ...)
```

formula	j.w.
data	j.w.
na.action	sposób postępowania z brakami w danych
method	u nas będzie to "class", ponieważ klasyfikujemy dane ze względu na zmienną typu factor
control	obiekt opisany niżej

```
rpart.control(minsplit=20, minbucket=round(minsplit/3), cp=0.01,
    xval=10, maxdepth=30, ...)
```

minsplit	minimalna liczba obserwacji w węźle, aby dokonać jego podziału
minbucket	minimalna liczba obserwacji w liściu, jeśli określony jest tylko jeden z parametrów *minsplit* i *minbucket*, druga z wartości zostaje dobrana automatycznie zgodnie ze wzorem $minsplit = minbucket * 3$
cp	parametr złożoności drzewa, żaden podział, który nie poprawia dostatecznie zdolności klasyfikacyjnych drzewa, nie jest wykonywany
xval	liczba kroswalidacji
maxdepth	j.w.

Do nadzorowania efektów funkcji tree służy natomiast:

```
tree(formula, data, na.action = na.pass, control =
    tree.control(nobs, ...), method = "recursive.partition",
    split = c("deviance", "gini"), model = FALSE,
     x = FALSE, y = TRUE, wts = TRUE, ...)
```

formula	j.w. — lewa strona (zmienna, którą klasyfikujemy) powinna być zadana wektorem numerycznym lub faktorem, a prawa składać się ze zmiennych (również typu liczbowego lub faktor), które R ma brać pod uwagę, dokonując podziałów węzłów
data	j.w.
na.action	j.w., domyślne *na.pass* przerzuca rekordy z brakującymi danymi do jak najniższych węzłów
control	obiekt *tree.control* opisany niżej
method	j.w.

```
tree.control(nobs, mincut = 5, minsize = 10, mindev = 0.01)
```

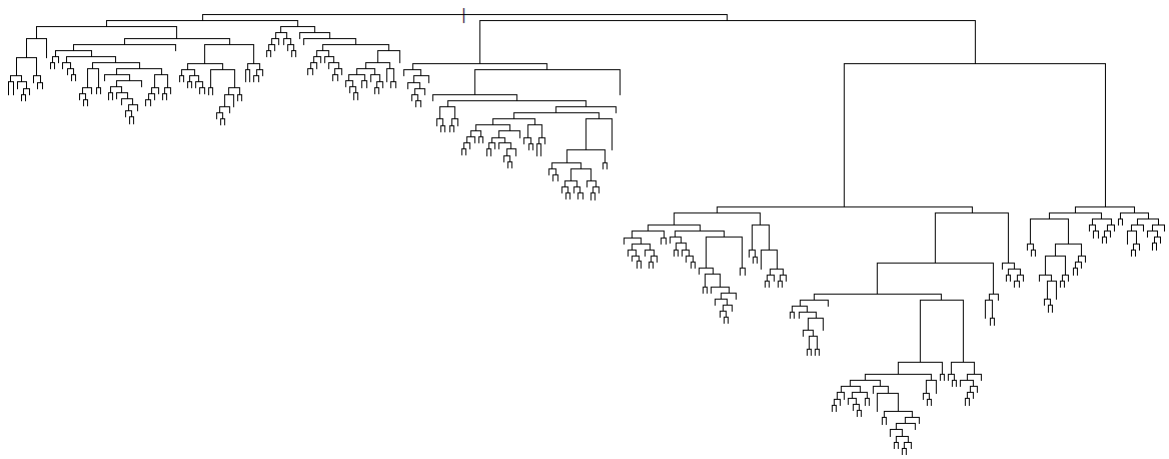
mincut	minimalna liczba obserwcaji w każdym węźle-dziecku, domyślnie=5
minsize	najmniejszy dopuszczalny rozmiar węzła, domyślnie =10
mindev	dewiancja wewnątrz węzła musi być przemnożona przez conajmniej taki współczynnik w stosunku do dewiancji w korzeniu, aby wykonać podział

W dalszej części tego rozdziału wykorzystuję dane, których używam i które dość szczegółowo opisałam w rozdziale trzecim. Zmienną ze względu, na którą będe tu klasyfikować jest MANSA - jest to wynik testu mierzącego jakość życia. Są trzy klasy: niski, średni i wysoki poziom zadowolenia z życia.

2.2. Możliwości pakietu *rpart*

Zobaczymy, jakie możliwości w zakresie drzew decyzyjnych daje nam pakiet rpart. Na początek wytrenujmy drzewo. Należy zauważyć, że domyślnie funkcja *rpart* ma już wprowadzone ograniczenia na rozbudowę drzewa, a więc jeśli chcemy uzyskać drzewo pełne, również należy skorzystać z funkcji *rpart.control*.

```
> tree.rpart <- rpart(MANSAd ~ Osrodek + tryb + cywilny + plec +
    dzieci + mieszka + edukacja + praca + bprs, method ="class",
    data=daneE, control = rpart.control(cp=0.0,minsplit=0,
    minbucket=0))
```



Rys.3. Drzewo pełne, jakie uzyskaliśmy dzięki powyższemu poleceniu.

Teraz ograniczmy rozbudowę.

```
> ptree.rpart <- rpart(MANSAd ~ Osrodek + tryb + cywilny + plec
    + dzieci + mieszka + edukacja + praca + bprs, method ="class",
    data=daneE, control=rpart.control(minsplit = 40, cp=0.01,
    maxdepth=5))
```

Wywołując powstały obiekt, dostaniemy inny niż na wykresie, wygodny (o ile drzewo nie jest zbyt duże) sposób analizowania kolejnych węzłów drzewa...

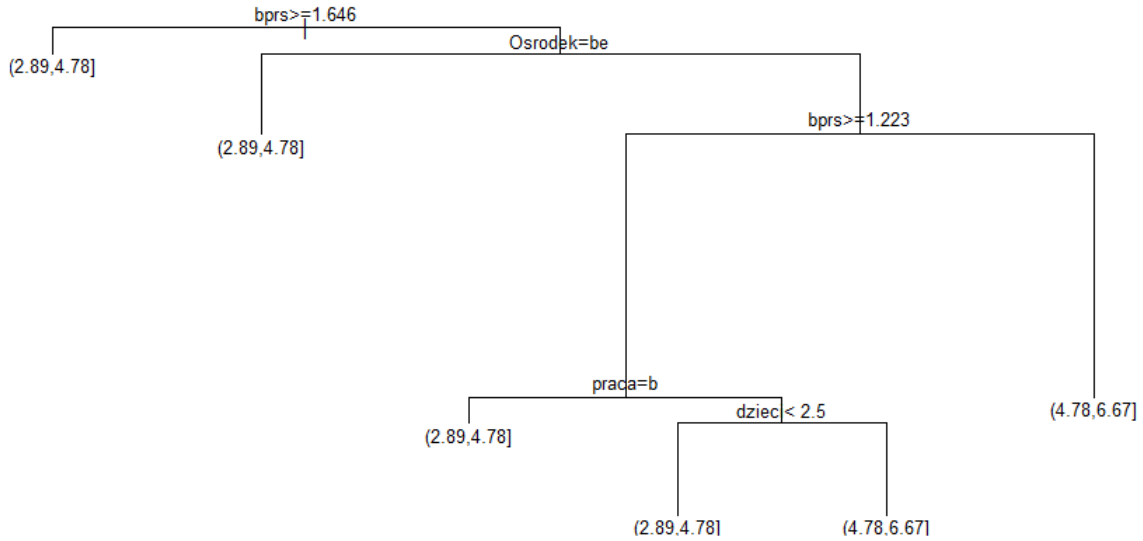
```
> ptree.rpart
n= 642

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

 1) root 642 276 (2.89,4.78]
                        (0.05763240 0.57009346 0.37227414)
   2) bprs>=1.645833 214  72 (2.89,4.78]
                        (0.12149533 0.66355140 0.21495327) *
   3) bprs< 1.645833 428 204 (2.89,4.78]
                        (0.02570093 0.52336449 0.45093458)
     6) Osrodek=London,Wroclaw 122  45 (2.89,4.78]
                        (0.04918033 0.63114754 0.31967213) *
     7) Osrodek=Dresden,Michalovce,Prague 306 152 (4.78,6.67]
                        (0.01633987 0.48039216 0.50326797)
      14) bprs>=1.223214 218 100 (2.89,4.78]
                         (0.02293578 0.54128440 0.43577982)
        28) praca=pracuje 91  35 (2.89,4.78]
                         (0.02197802 0.61538462 0.36263736) *
        29) praca=nie pracuje 127  65 (2.89,4.78]
                         (0.02362205 0.48818898 0.48818898)
          58) dzieci< 2.5 114  54 (2.89,4.78]
                         (0.01754386 0.52631579 0.45614035) *
```

```
      59) dzieci>=2.5 13    3 (4.78,6.67]
                    (0.07692308 0.15384615 0.76923077) *
   15) bprs< 1.223214 88   29 (4.78,6.67]
                    (0.00000000 0.32954545 0.67045455) *
```

... co nie stoi na przeszkodzie, by przyjrzeć się wykresowi. Polecenia:

```
> plot(ptree.rpart)
> text(ptree.rpart, cex=.6)
```

dają następujący efekt:



Rys.4. Drzewo powstałe w wyniku narzucenia warunków stopu.

W wyniku wywołania zbudowanego drzewa widzimy, że dokonanych zostało 5 podziałów (tyle, ile wynosiło ograniczenie parametrem *maxdepth*) oraz że najmniej liczny węzeł, jaki uległ podziałowi, składał się ze 127 elementów.

Spójrzmy, co się dzieje w najdalszych węzłach drzewa pełnego. Posłuży nam do tego polecenie *summary(tree.rpart)*. Umieszczam tu jedynie początek i koniec odpowiedzi programu R, gdyż wystarczy to do zrozumienia działania tej funkcji w zastosowaniu do drzew decyzyjnych.

```
> summary(tree.rpart)
Call:
rpart(formula = MANSAd ~ Osrodek + tryb + cywilny + plec + dzieci +
    mieszka + edukacja + praca + bprs, data = daneE, method = "class",
    control = rpart.control(cp = 0, minsplit = 0, minbucket = 0))
  n= 642

           CP nsplit  rel error    xerror       xstd
1  0.036231884      0 1.000000000 1.0000000 0.04544839
2  0.014492754      3 0.891304348 0.9963768 0.04542791
3  0.010869565      8 0.807971014 0.9782609 0.04531863
4  0.007246377     11 0.775362319 1.0253623 0.04557890
```

```
5  0.006038647     19 0.717391304 1.0362319 0.04562800
6  0.005434783     22 0.699275362 1.0507246 0.04568712
7  0.004830918     34 0.634057971 1.0724638 0.04576224
8  0.003623188     53 0.528985507 1.0724638 0.04576224
9  0.002717391    104 0.344202899 1.1413043 0.04589365
10 0.002415459    121 0.289855072 1.1376812 0.04589075
11 0.002173913    136 0.253623188 1.1376812 0.04589075
12 0.001811594    141 0.242753623 1.1557971 0.04590077
13 0.001449275    234 0.061594203 1.1557971 0.04590077
14 0.001207729    246 0.043478261 1.1847826 0.04589365
15 0.000000000    278 0.003623188 1.1847826 0.04589365

Node number 1: 642 observations,    complexity param=0.03623188
  predicted class=(2.89,4.78]   expected loss=0.4299065
    class counts:    37   366   239
   probabilities: 0.058 0.570 0.372
  left son=2 (214 obs) right son=3 (428 obs)
  Primary splits:
      bprs    < 1.645833 to the right, improve=12.057630, (0 missing)
      Osrodek splits as  RLRRL,        improve= 9.097109, (0 missing)
      cywilny splits as  LR,           improve= 2.457444, (0 missing)
      dzieci  < 0.5      to the left,  improve= 2.321805, (0 missing)
      mieszka splits as  LR,           improve= 1.691550, (0 missing)
  Surrogate splits:
      dzieci   < 4.5      to the right, agree=0.668, adj=0.005,
                                                    (0 split)
      edukacja < 18.5     to the right, agree=0.668, adj=0.005,
                                                    (0 split)

...

Node number 954854: 2 observations,    complexity param=0.001811594
  predicted class=(2.89,4.78]   expected loss=0.5
    class counts:     0     1     1
   probabilities: 0.000 0.500 0.500
  left son=1909708 (1 obs) right son=1909709 (1 obs)
  Primary splits:
      bprs < 1.333333 to the right, improve=1, (0 missing)

Node number 954855: 4 observations
  predicted class=(4.78,6.67]   expected loss=0
    class counts:     0     0     4
   probabilities: 0.000 0.000 1.000

Node number 1909708: 1 observations
  predicted class=(2.89,4.78]   expected loss=0
    class counts:     0     1     0
   probabilities: 0.000 1.000 0.000
```

```
Node number 1909709: 1 observations
  predicted class=(4.78,6.67]    expected loss=0
    class counts:     0     0     1
   probabilities: 0.000 0.000 1.000
```

Jak to interpertować?

numer węzła	liczba obserwacji w węźle		
klasa, do której węzeł został zaklasyfikowany	stosunek źle zaklasyfikowanych obserwacji w węźle do dobrze zaklasyfikowanych		
liczebność klas w węźle	niski	średni	wysoki
stosunek liczby obserwacji z danej klasy do liczby wszystkich obserwacji w tym węźle	niski/wszystkie	średni/wszystkie	wysoki/wszystkie

Słowa: niski, średni i wysoki odnoszą się do wyników uzyskanych w teście MANSA.

Trudno oczekiwać, żeby takie drzewo było rzeczywiście przydatne w zadaniu dyskryminacyjnym, skoro o przynależności do danej klasy decyduje czasem tylko jedna obserwacja. Takie postępowanie przeczy założeniom klasyfikacji, a więc łączeniu obiektów o podobnych właściwościach w większe grupy.

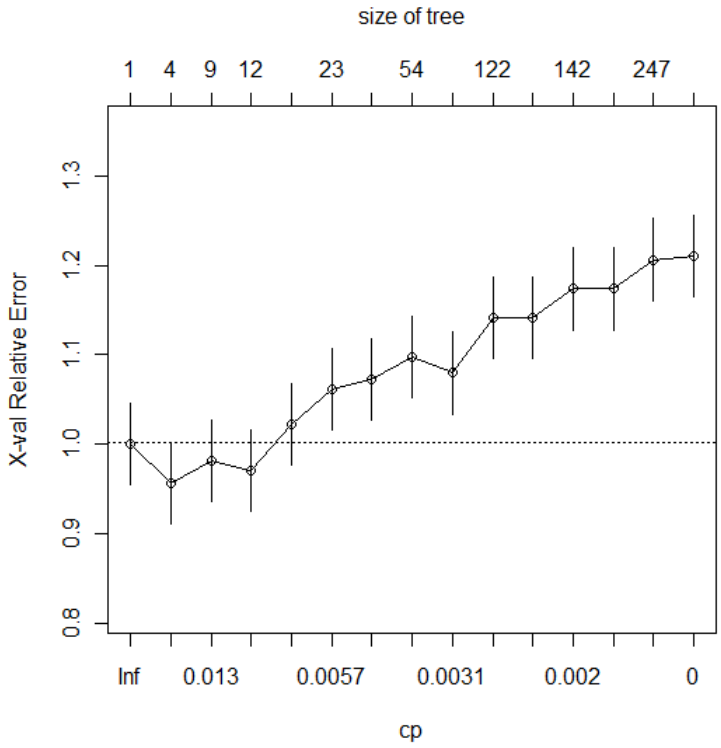
Przydatną jest też funkcja *printcp*. Możemy dzięki niej uzyskać tabelę wskazującą, jakie drzewa dostaniemy dla określonego *cp* - parametru złożoności (patrz: 1.3.2). Nietrudno zauważyć, że również funkcja *summary* dawała informację o tym parametrze. Nie pokazywała natomiast, które ze zmiennych opisujących dane rzeczywiście zostały użyte do budowy drzewa. Spójrzmy zatem, jak działa ta funkcja.

```
printcp(tree.rpart)

Classification tree:
rpart(formula = MANSAd ~ Osrodek + tryb + cywilny + plec + dzieci +
    mieszka + edukacja + praca + bprs, data=daneE, method="class",
    control = rpart.control(cp = 0, minsplit = 0, minbucket = 0))

Variables actually used in tree construction:
[1] bprs    cywilny dzieci  edukacja mieszka Osrodek plec    praca
[9] tryb

Root node error: 276/642 = 0.42991

n= 642

          CP nsplit rel error  xerror     xstd
1  0.0362319      0 1.0000000 1.00000 0.045448
2  0.0144928      3 0.8913043 0.92029 0.044891
3  0.0108696      8 0.8079710 0.93116 0.044980
4  0.0072464     11 0.7753623 0.97464 0.045295
5  0.0060386     19 0.7173913 1.01449 0.045526
6  0.0054348     22 0.6992754 1.04348 0.045658
7  0.0048309     34 0.6340580 1.04710 0.045673
8  0.0036232     53 0.5289855 1.06884 0.045751
```

9	0.0027174	104	0.3442029	1.10145	0.045837
10	0.0024155	121	0.2898551	1.13043	0.045884
11	0.0021739	136	0.2536232	1.15942	0.045901
12	0.0018116	141	0.2427536	1.19203	0.045887
13	0.0014493	234	0.0615942	1.18841	0.045891
14	0.0012077	246	0.0434783	1.18478	0.045894
15	0.0000000	278	0.0036232	1.18116	0.045896

Widzimy tutaj, że dla $cp \in [0, 0.0012077)$ wybrane zostanie drzewo, mające 278 podziałów (drzewo pełne), dla $cp \in [0.0012077, 0.0014493)$ drzewo powstałe w wyniku 246 podziałów, itd., aż do $cp \in [0.0362319, \infty)$, które dają w efekcie drzewo bez podziałów (sam korzeń).

Możemy również uzyskać wykres cp obrazujący powyższą tabelkę. Służy do tego funkcja *plotcp()*.

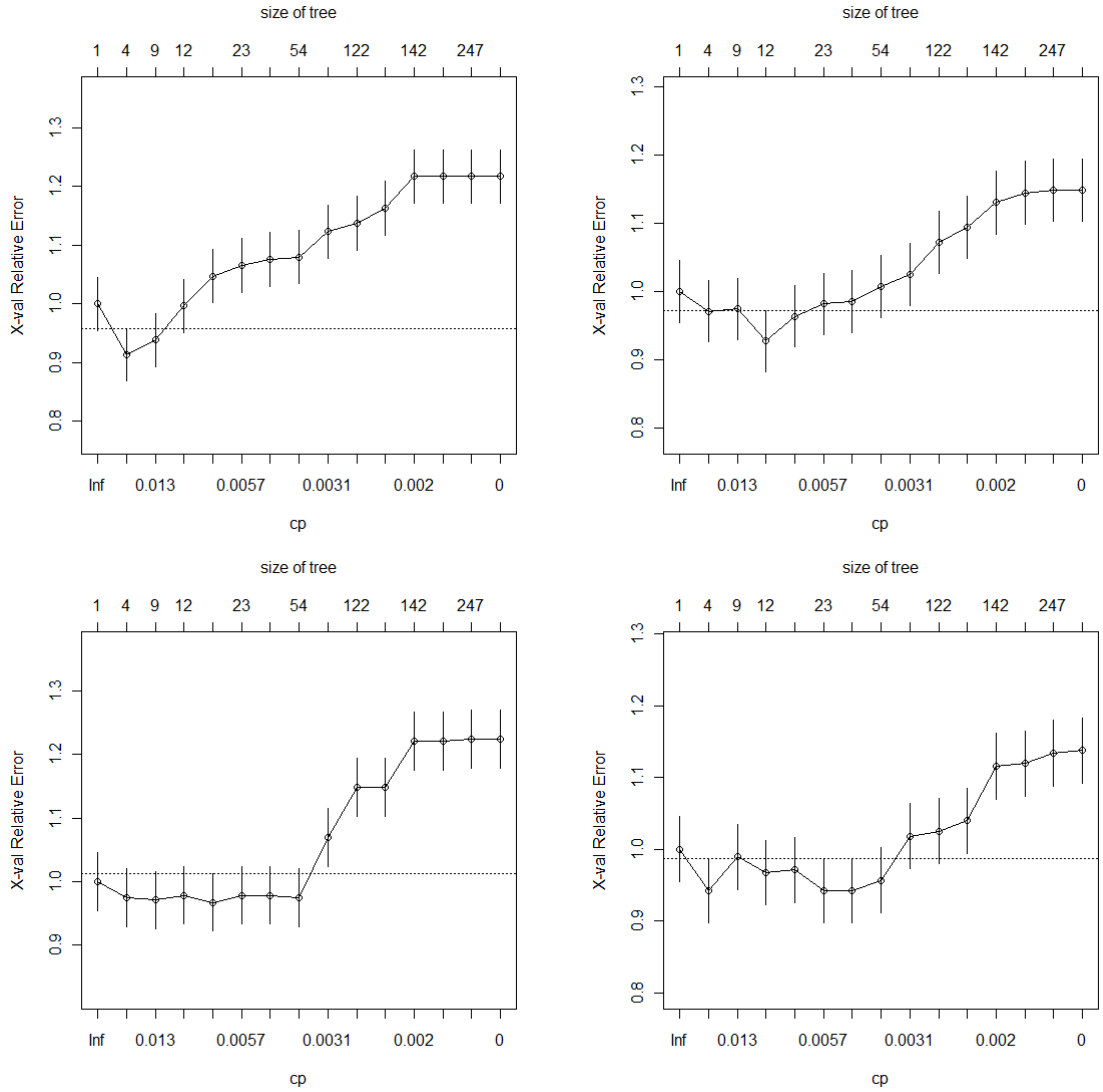


Rys.5. Wykres cp.

Wykres taki może być pomocny przy wyborze takiego cp, aby drzewo powstałe w wyniku przycinania miało jak najlepsze zdolności klasyfikacyjne. Zgodnie z algorytmem opartym na kryterium kosztu-złożoności (1.3.2) powinniśmy wybrać drzewo, które daje najmniejszy błąd klasyfikacji. Kierując się natomiast regułą 1 SE, o której również była mowa w paragrafie 1.3.2, jako optymalne drzewo, wybrane zostać powinno najmniejsze spośród tych, których sumaryczny ułamek błędnych klasyfikacji ($Q_s(T_j)$) jest odległy o nie więcej niż jedno odchylenie standardowe od minimum ułamka błędnych klasyfikacji ($Q_s(T_{j_0})$). Funkcja *plotcp* domyślnie rysuje linię na wysokości jednego odchylenia standardowego powyżej tej minimalnie uzyskanej wartości, wobec czego pozostaje wybrać najmniejsze z drzew, dla których wykres błędu znajduje się pod kreską. Na powyższym rysunku widać jednak, że chcąc wybrać drzewo według takiej reguły, wybierzemy sam korzeń. To by oznaczało, że wszystkie obserwacje ze zbioru uczącego są tak podobne, że klasyfikowanie ich nie da lepszych efektów niż powie-

dzenie na wstępie, że wszyscy pacjenci są średnio zadowoleni z życia. Innymi słowy oznacza to, że tak postawione zadanie klasyfikacji nie ma sensu. Czy jednak rzeczywiście tak jest?

Należy zauważyć, że uzyskana tabelka jest bezpośrednim wynikiem kroswalidacji, którą opisywałam w 1.3.2, a więc jest zdarzeniem losowym. Sensownym wydaje się sprawdzenie, czy zawsze (a w zasadzie, jak często) dostajemy taki wynik. Na 25 prób analogiczny efekt uzyskałam 10 razy. Dostałam między innymi następujące wykresy:



Rys. 6. Przykładowe wykresy cp dla drzewa pełnego.

Dało się również zauważyć, że w większości przypadków najmniejszy błąd uzyskiwaliśmy dla rozmiaru drzewa 4 lub 9.

Skoro przyjrzeliśmy się już drzewu pełnemu i przeanalizowaliśmy zdolności klasyfikacyjne drzewa w zależności od jego rozmiaru i wybranego cp, możemy przejść do wyboru odpowiedniego rozmiaru drzewa.

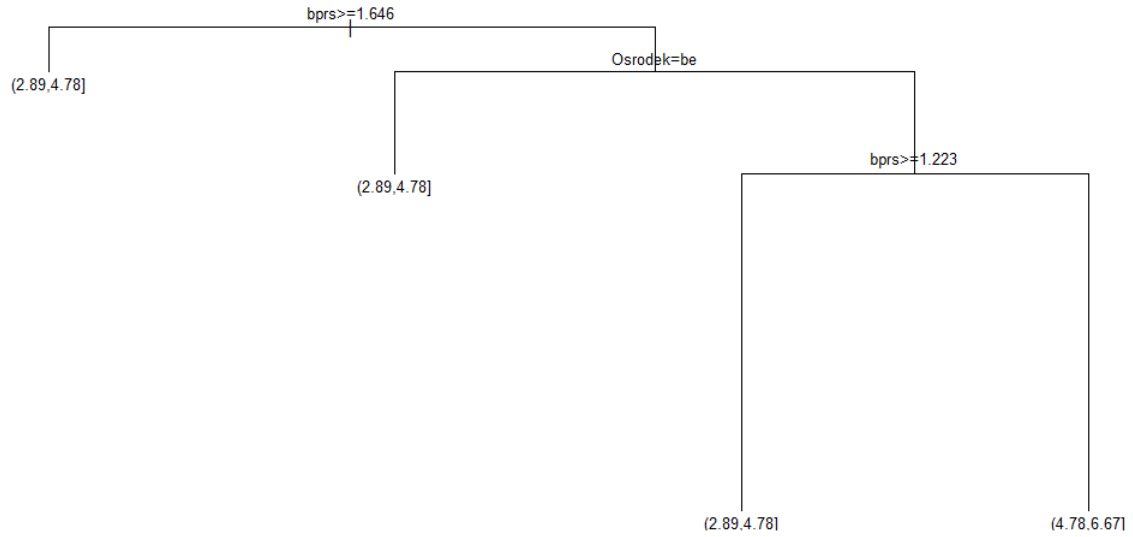
Pokazaliśmy, jak z góry ograniczyć rozbudowę drzewa z użyciem pakietu rpart. Czas sprawdzić, jak wygląda przycinanie drzewa. Funkcja *prune.rpart()* działa w oparciu o algorytm kosztu-złożoności.

```
> plot(prune.rpart(tree.rpart,cp=0.05))
```

```
Error in plot.rpart(prune.rpart(tree.rpart, cp = 0.05)) :
  fit is not a tree, just a root
```

Powyższy wynik pokazuje, co się dzieje dla zbyt dużego parametru złożoności. Natomiast niżej przedstawiam drzewo przycięte dla cp wybranego na podstawie wcześniejszej analizy (3.2.1). Przypomnę, że najmniejszy błąd w podanym przykładzie działania funkcji *printcp* uzyskaliśy dla $cp \in [0.0144928, 0.0362319)$.

```
> pr.tree.rpart=prune.rpart(tree.rpart,cp= 0.02)
> plot(pr.tree.rpart)
> text(pr.tree.rpart,cex=0.8)
```



Rys.7. Efekt przycięcia drzewa pełnego funkcją *prune.rpart()*.

Aby sprawdzić, gdzie powstałe drzewo zaklasyfikuje nową obserwację, możemy użyć funkcji *predict()*. Spójrzmy, jak jej używać.

```
> predict(ptree.rpart,daneE)
  (0.994,2.89] (2.89,4.78] (4.78,6.67]
1   0.00000000   0.3295455   0.6704545
2   0.01754386   0.5263158   0.4561404
3   0.12149533   0.6635514   0.2149533
4   0.02197802   0.6153846   0.3626374
5   0.12149533   0.6635514   0.2149533
```

Dla każdej obserwacji dostaliśmy tu prawdopodobieństwo zostania zaklasyfikowanym do danej klasy (liczby w wierszach sumują się do jedynki z dokładnością do liczby miejsc po przecinku). Oczywiście dla drzewa pełnego dostajemy tabelę zer i jedynek.

```
> predict(tree.rpart,data=daneE)
  (0.994,2.89] (2.89,4.78] (4.78,6.67]
1            0        0.00        1.00
2            0        1.00        0.00
```

```
3                0          0.00          1.00
4                0          1.00          0.00
5                0          1.00          0.00
6                0          0.00          1.00
7                0          1.00          0.00
8                0          1.00          0.00
9                0          1.00          0.00
10               0          1.00          0.00
...
```

Inną ciekawą funkcją jest *xpred.rpart()*, która pozwala zobaczyć, do jakiej klasy zostały przypisane przez drzewo poszczególne obserwacje w wyniku zastosowania kroswalidacji (a więc obserwacje ze zbioru uczącego). Argument *xval=...* określa liczbę zbiorów kroswalidacyjnych, jaka ma zostać użyta.

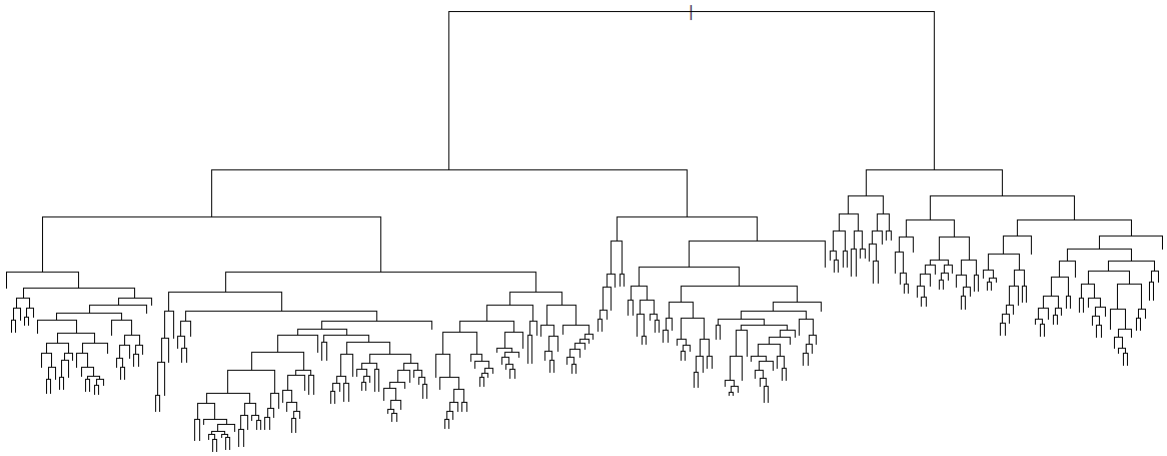
```
> xpred.rpart(ptree.rpart, xval=10)
     0.51811594 0.02291506 0.01203859
1             2          3          3
2             2          2          2
3             2          2          2
4             2          2          2
5             2          2          2
6             2          3          3
...
```

Widzimy, jak zaklasyfikowanych zostało pierwszych sześciu pacjentów. Trzy liczby w nagłówkach kolumn to współczynniki cp, dla których przeprowadzono kroswalidację. 2 oznacza tu średni wynik w teście MANSA, natomiast 3 — wysoki.

2.3. Pakiet *tree*

Zajmijmy się teraz możliwościami pakietu *tree*.

```
> tree.tree <- tree(MANSAd ~ Osrodek + tryb + cywilny + plec + dzieci
  + mieszka + edukacja + praca + bprs, method ="class", data=daneE,
  control=tree.control(642,minsize=2,mincut=1,mindev=0.0))
```



Rys.8. Drzewo pełne uzyskane funkcją *tree*.

Do przycinania drzew klasyfikacyjnych w tym pakiecie służy funkcja *prune.tree* lub jej uproszczony wariant *prune.misclass*. Uproszczony, ponieważ ma domyślnie ustawiony parametr *method="misclass"*, gdzie parametr ten oznacza miarę różnorodności, jakiej chcemy użyć (patrz: 1.2.1). Argument *k=...* pozwala na wybór parametru złożoności i uzyskać drzewo optymalne dla tego parametru (k jest tu odpowiednikiem *cp* z 2.2 oraz α z 1.3.2). Natomiast dzięki parametrowi *best=...* możemy wybrać rozmiar drzewa, jaki chcemy uzyskać. Jeśli nie podamy żadnego z tych argumentów dostaniemy ciąg drzew optymalnych ze względu na poszczególne wartości k (przykład poniżej).

```
> ptree.tree<-prune.misclass(tree.tree)
> ptree.tree
$size
 [1] 265 264 255 250 212 202 163 139  54  48  39  35  32  27  25  17
 14   4   1

$dev
 [1]   1   1   4   6  25  31  57  75 160 167 178 183 187 194 197 213
 220 246 276

$k
 [1]       -Inf  0.0000000  0.3333333  0.4000000  0.5000000
 0.6000000  0.6666667  0.7500000  1.0000000  1.1666667  1.2222222
 1.2500000  1.3333333  1.4000000 1.5000000  2.0000000  2.3333333
 2.6000000 10.0000000

$method
[1] "misclass"

attr(,"class")
[1] "prune"         "tree.sequence"
```

Przy pomocy argumentu *newdata=...* możemy przycinać drzewo z wykorzestaniem nowych danych.

2.4. Kilka przydatnych funkcji pakietu *maptree*

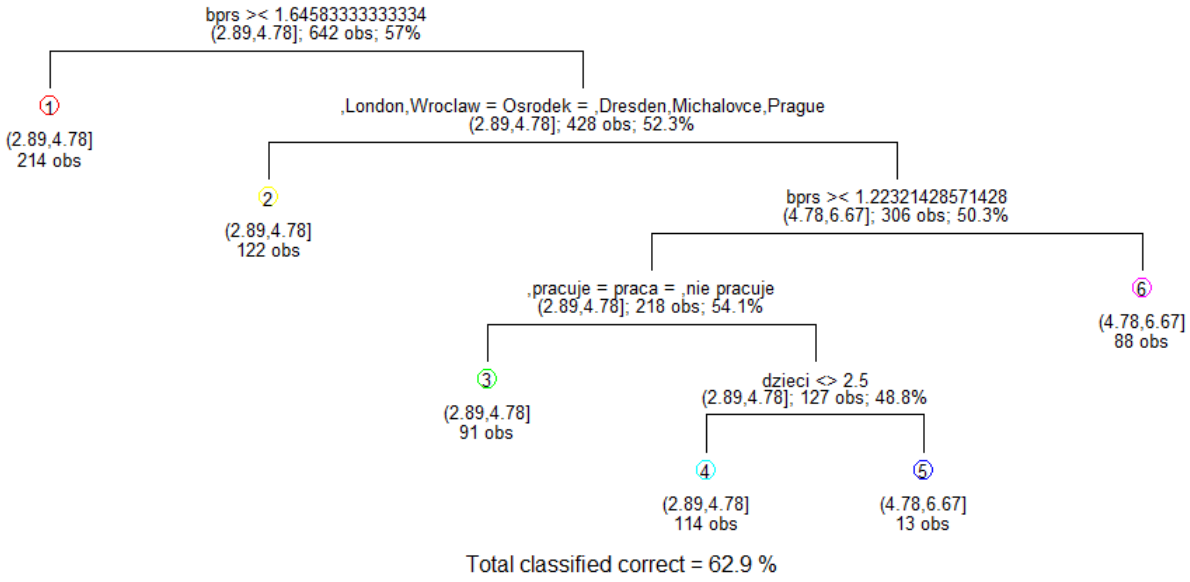
Przedstawiona niżej funkcja służy do przycinania drzewa powstałego z użyciem funkcji *rpart()*. Oprócz drzewa do przycięcia w argumentach podajemy, jakiego kryterium użyć — może to być liczba podziałów węzłów (*best=...*) lub cp, dla jakiego minimalizowana ma być funkcja kosztu–złożoności (*cp=...*).

```
> clip.rpart(tree.rpart, best=4)
```

Funkcja...

```
> draw.tree(ptree.rpart, nodeinfo=TRUE, print.levels=TRUE, cex=.8)
```

...narysuje drzewo podane jako argument (obiekt klasy *rpart* lub *tree*). Poniżej efekt.



Rys.9. Drzewo pełne uzyskane funkcją *tree*.

Zaletą tego wykresu w stosunku do wykresu drzewa wywołanego funkcją *plot()* jest większa ilość informacji, jakie zawiera. Mamy tu podane, oprócz kryterium podziału w węzłach i klasyfikacji liści, również liczby obserwacji w poszczególnych węzłach i informację, jaka część zbioru została trafnie zaklasyfikowana.

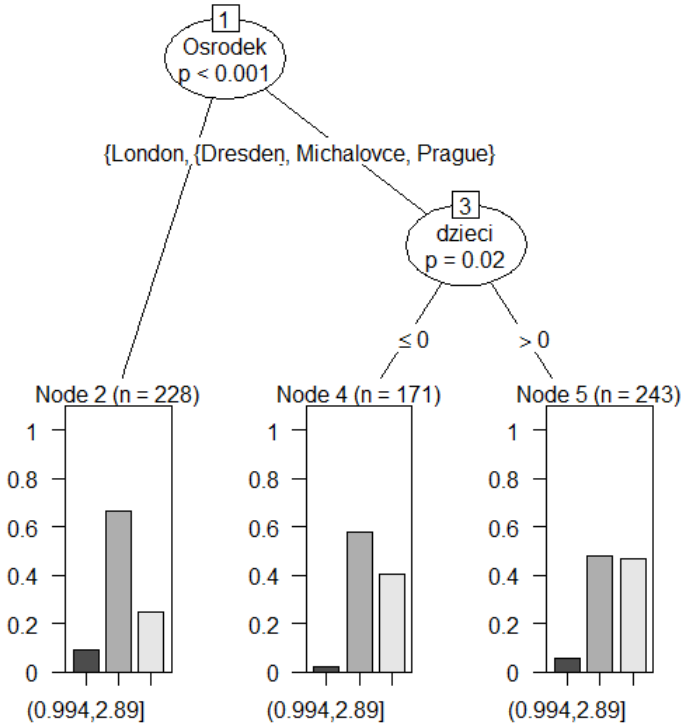
Funkcja *goup.tree()* pozwala zobaczyć dokładnie, które obserwacje znalazły się w którym węźle.

```
> k=group.tree(ptree.rpart)
> k
  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17
18  19  20
  6   4   1   3   1   6   3   1   3   1   3   4   4   6   6   1   6
6   1   4
...
```

2.5. Pakiet *party*

Ciekawe od strony graficznej efekty można uzyskać dzięki funkcji *ctree*.

```
> tree.ctree<-ctree(MANSAd ~ Osrodek + tryb + cywilny + plec
+ dzieci + mieszka + edukacja + praca, data=daneE)
```



Rys.10. Drzewo uzyskane funkcją *ctree*.

Jak widać funkcja ta pozwala na wygodne porównania liczebności różnych klas w poszczególnych liściach.

Wywołanie drzewa nie daje już jednak informacji, jakiego atrybutu użyto do podziału na konkretnym etapie.

```
> tree.ctree

        Conditional inference tree with 3 terminal nodes

Response:  MANSAd
Inputs:  Osrodek, tryb, cywilny, plec, dzieci, mieszka, edukacja, praca
Number of observations:  642

1) Osrodek == {London, Wroclaw}; criterion = 0.999, statistic = 31.705
  2)*  weights = 228
1) Osrodek == {Dresden, Michalovce, Prague}
  3) dzieci <= 0; criterion = 0.98, statistic = 11.953
    4)*  weights = 171
  3) dzieci > 0
    5)*  weights = 243
```

Rozdział 3

Przycinanie drzew z użyciem pakietu R na podstawie rzeczywistych danych

Przedstawiony zostały już aspekt teoretyczny przycinania drzew klasyfikacyjnych, jak również przydatne polecenia pakietu R. Najwyższy czas połączyć informacje z obydwu rozdziałów i zobaczyć, jak rzecz wygląda w praktyce. W tej części postaram się pokazać, jak działa przycinanie drzew na rzeczywistych danych.

3.1. Opis danych

Aby dobrze rozumieć, co się dzieje w tym rozdziale, trzeba najpierw poznać dane, na jakich będziemy pracować i wobec tego pokrótce je teraz omówię.

Dane te pochodzą z projektu o nazwie EDEN.[1] Projekt ten miał na celu porównanie dwóch sposobów leczenia pacjentów z problemami natury psychiatrycznej. Ze względu na duże koszty leczenia w placówkach zamkniętych chciano sprawdzić, czy przypadkiem leczenie w tzw. trybie dziennym (pacjent przychodzi codziennie na kilka godzin leczenia, a potem wraca do domu) nie przynosi porównywalnych rezultatów.
W tym celu zebrano dane 642 pacjentów z pięciu ośrodków w Europie: Londyn, Drezno, Praga, Michalovce (Słowacja) i Wrocław.

Dane zawierają informacje takie jak:

- ośrodek, w którym leczono danego pacjenta (jeden z pięciu wyżej wymienionych),
- tryb leczenia – dzienny lub stacjonarny,
- płeć,
- stan cywilny chorego – w relacji lub samotnie,
- liczba dzieci,
- liczba lat edukacji,
- informacja, czy mieszka z kimś, czy samotnie,

[1] Więcej informacji dostępnych na stronie projektu: http://www.edenstudy.com/pl

- informacja, czy pracuje,
- wyniki dwóch testów, z których jeden mierzył poziom nasilenia objawów psychopatologicznych, a drugi badał zadowolenie z życia pacjenta.

3.1.1. Testy, jakich użyto w badaniach pacjentów

MANSA

Manchester Short Assessment of Quality of Life[2] jest sześciopunktową skalą powstałą w 1999 roku używaną do oceny jakości życia. Jest ona powszechnie używana przez psychologów. Składa się z 16 pytań. W większości są to pozycje oceniające w sposób całkowicie subiektywny satysfakcję pacjenta z jego własnego życia jako całości oraz z poszczególnych jego aspektów takich jak zatrudnienie czy relacje rodzinne, itp. Obiektywne pozycje skali dotyczą natomiast na przykład istnienia bliskich przyjaciół lub bycia ofiarą przemocy fizycznej czy podmiotem oskarżenia o popełnienie przestępstwa.

BPRS

Brief Psychiatric Rating Scale jest testem używanym przez klinicystów do oceny poziomu zaburzeń psychopatologicznych. Test składa się z 24 pozycji wymieniających konkretne symptomy, których nasilenie należy określić wybierając odpowiednią liczbę od 1 (not assessed) do 7 (extremely severe). Cała skala składa się z 4 podskal dotyczących:

- depresji,
- zachowań maniakalnych,
- objawów pozytywnych, a więc występujących u badanego, chociaż u zdrowego człowieka nie występują (np. halucynacje, rozkojarzenie myślenia, dziwaczne zachowania),
- objawów negatywnych, a więc pewnych dysfunkcji w stosunku do osób zdrowych (np. apatia, wycofanie emocjonalne, brak płynności w rozmowach).

Podskale te łącznie dają ocenę ilościową zaburzeń, pozwalając stwierdzić, na ile dana osoba wykazuje objawy psychopatologiczne, natomiast rozpatrywane oddzielnie dają ocenę jakościową, wskazując, w jakim zakresie dana osoba ma problemy. [3]

3.1.2. Bardziej szczegółowy opis danych

Dysponując wymienionymi zmiennymi, możemy postawić wiele różnych problemów klasyfikacyjnych. Ja postaram się odpowiedzieć na pytanie: *Które zmienne i w jaki sposób wpływają na jakość życia badanych?*

[2] Abstrakt dotyczący testu można przeczytać na stronie:
http://isp.sagepub.com/content/45/1/7.short
Pewne informacje są też umieszczone na stronie projektu EDEN:
http://www.edenstudy.com

[3] Dokładny opis objawów znaleźć można pod adresem:
http://www.public-health.uiowa.edu/icmha/outreach/documents/BPRS_expanded.PDF

Podstawowe statystyki

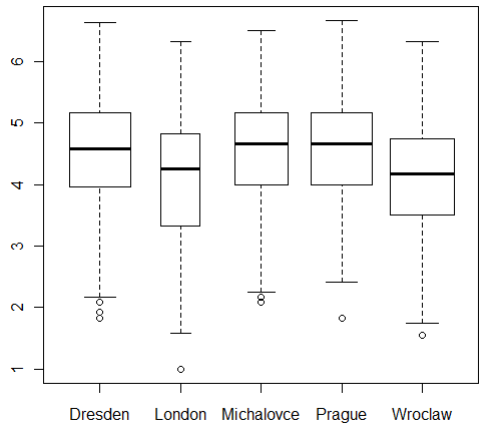
```
Osrodek        System      Dzienni.stacjonarni     Stan.cywilny
Dresden    :147   wschod:433   dzienni     :342      samotnie :356
London     : 62   zachod:209   stacjonarni:300      w relacji:286
Michalovce:114
Prague     :153
Wroclaw    :166

   plec          dzieci           mieszka.samotnie.z.kims
kobieta   :403   Min.    : 0.000   mieszka samotnie:110
mezczyzna:239   1st Qu.: 0.000   mieszka z kims  :532
                Median : 1.000
                Mean   : 1.196
                3rd Qu.: 2.000
                Max.   :15.000

liczba.lat.edukacji      czy.pracuje
Min.   : 5.00        nie pracuje:428
1st Qu.:10.00        pracuje    :214
Median :12.00
Mean   :12.33
3rd Qu.:14.00
Max.   :23.00
```

Tak natomist wyglądaja statystyki wyników testów psychometrycznych.

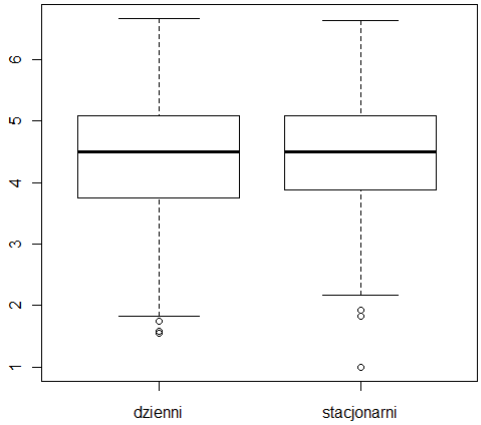
```
BPRS.Manic.Excitement BPRS.Negative.Symptoms BPRS.Positive.Symptoms
Min.   :1.000         Min.   :1.000          Min.   :1.000
1st Qu.:1.000         1st Qu.:1.000          1st Qu.:1.000
Median :1.167         Median :1.500          Median :1.200
Mean   :1.305         Mean   :1.587          Mean   :1.372
3rd Qu.:1.333         3rd Qu.:2.000          3rd Qu.:1.400
Max.   :5.000         Max.   :5.250          Max.   :5.400


BPRS.Depression.Anxiety  BPRS.Average      MANSA
Min.   :1.000            Min.   :1.000   Min.   :1.000
1st Qu.:1.500            1st Qu.:1.250   1st Qu.:3.822
Median :2.000            Median :1.500   Median :4.500
Mean   :2.212            Mean   :1.565   Mean   :4.410
3rd Qu.:2.750            3rd Qu.:1.792   3rd Qu.:5.083
Max.   :6.000            Max.   :3.667   Max.   :6.667
```

Które zmienne korelują z wynikiem testu MANSA?

Ciekawe może być sprawdzenie, jak się mają do siebie atrybuty wybierane przez drzewo do podziału i zmienne korelujące z jakością życia.



Rys.11. MANSA w poszczególnych ośrodkach.



Rys. 12. MANSA w zależności od trybu leczenia.

Test χ^2:

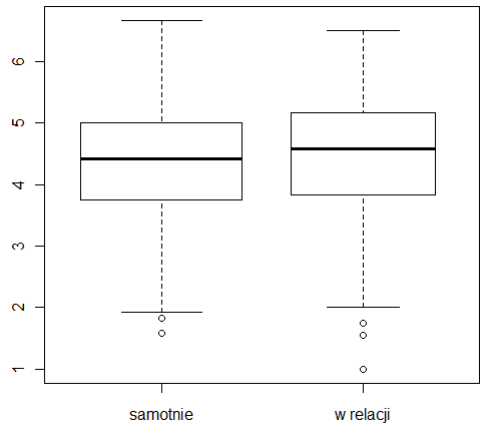
```
data:  MANSA and Osrodek
X-squared = 415.3121, df = 356,
p-value = 0.01639
```

Widzimy, że we Wrocławiu i Londynie wyniki testu MANSA są przesunięte w dół. Zależność ta jest istotna statystycznie.

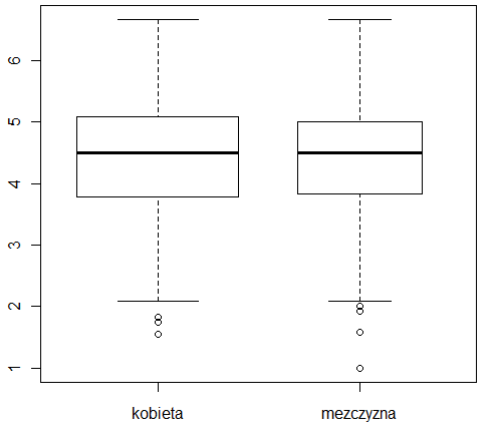
Test χ^2:

```
data:  MANSA and tryb
X-squared = 86.4644, df = 89,
p-value = 0.5564
```

Rozkłady zmiennej MANSA w zależności od trybu nie mają znacznych różnic na wykresie. Brak zależności potwierdza test.



Rys. 13. MANSA a fakt bycia w związku.



Rys. 14. MANSA w zależności od płci.

Test χ^2:

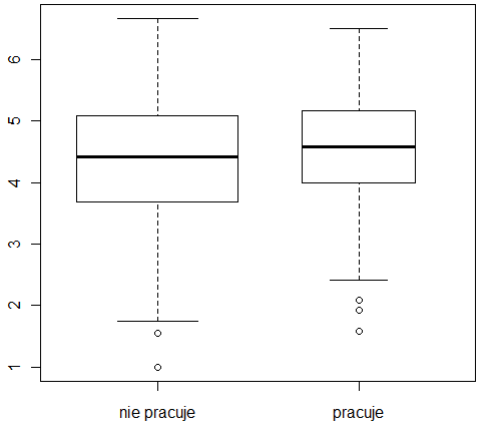
```
data:  MANSA and cywilny
X-squared = 103.1884, df = 89,
p-value = 0.1443
```

Na podstawie wykresu wydaje się, że osoby pozostające w związku są nieco bardziej zadowolone z życia, jednak test istotności tego nie potwierdza.

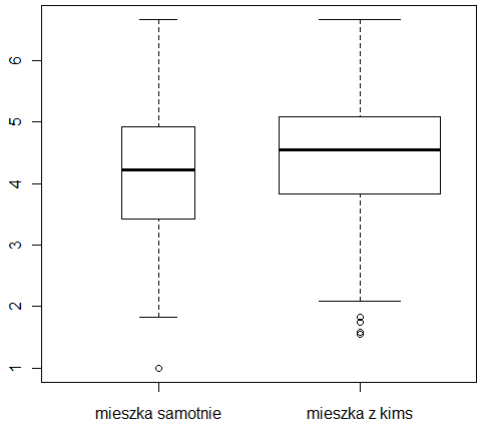
Test χ^2:

```
data:  MANSA and plec
X-squared = 77.3877, df = 89,
p-value = 0.8053
```

Płeć nie ma wpływu na ocenę jakości życia. Potwierdza to zarówno wykres, jak i istotność testu χ^2.



Rys. 15. MANSA w zależności od tego, czy pacjent pracuje.



Rys. 16. MANSA u pacjentów mieszkających samotnie\z kimś.

Test χ^2:

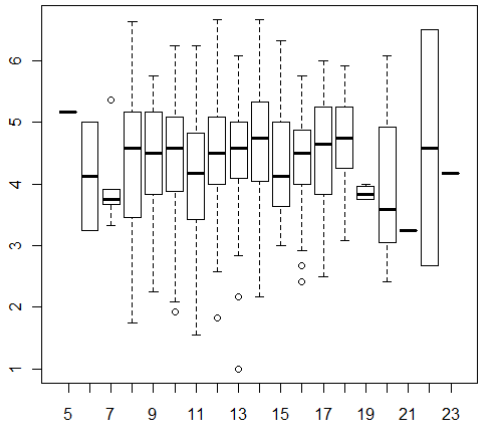
```
data:  MANSA and praca
X-squared = 85.4972, df = 89,
p-value = 0.5855
```

Mimo, że wykres może sugerować delikatną zależność jakości życia od posiadania pracy w badanej próbie, test tego nie potwierdza.

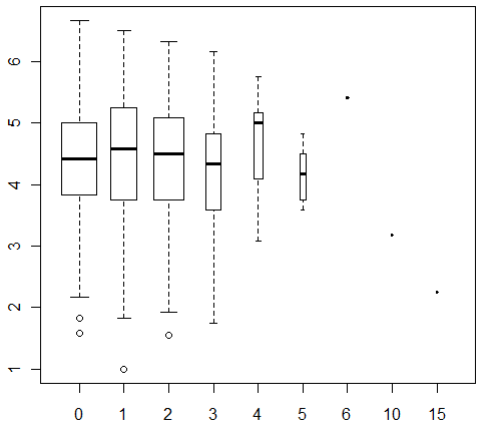
Test χ^2:

```
data:  MANSA and mieszka
X-squared = 135.7215, df = 89,
p-value = 0.00105
```

Z rysunku można odczytać, że być może osoby mieszkające samotnie mają niższe wyniki w teście MANSA. Test to potwierdza.



Rys. 17. MANSA a liczba lat edukacji (lata edukacji na dolnej osi).



Rys. 18. MANSA w zależności od liczby dzieci pacjenta.

Test χ^2:

```
data:  MANSA and edukacja
X-squared = 1729.499, df = 1602,
p-value = 0.01366
```

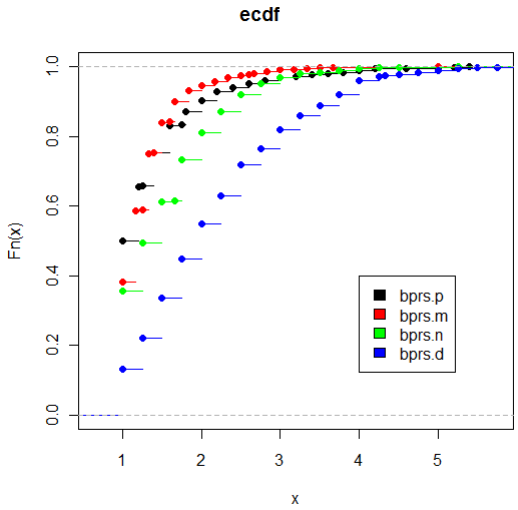
Wykres nie wskazuje wyraźnej zależności. Test istotności współczynnika Pearsona dał p-wartość równą 0.3783, a więc dał wynik inny niż test χ^2. Współczynnik korealcji=0.03483059, a więc nawet jeśli zależność jest, nie jest ona duża.

Test χ^2:

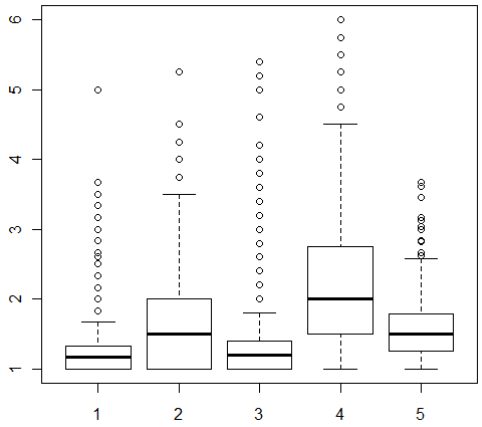
```
data:  MANSA and dzieci
X-squared = 1041.159, df = 712,
p-value = 8.498e-15
```

Brak wyraźnej tendencji, ale widzimy, że dla osób, które w ogóle mają dzieci jakość życia stopniowo maleje (wyjątkiem są osoby z czwórką dzieci - jest ich 17). Czy oznacza to rzeczywistą zależność? P-artość wskazuje na jej istnienie, ale test współczynnika Pearsona daje znów inny wynik: cor=-0.06533136, p-wartość= 0.09815. Wniosek jest analogiczny, jak w przypadku edukacji.

Przyjrzyjmy się teraz zmiennym określającym poziom zaburzeń psychicznych. Wyniki testu BPRS wśród badanych mają następujący rozkład.



Ry. 19. Dystrybuanta emipryczna dla podskal BPRS.



Rys. 20. Wykresy *boxplot* dla podskal BPRS - kolejno: bprs.m, bprs.n, bprs.p, bprs.d, średnia BPRS.

Do zbadania, czy wpływają one na jakość życia (intuicyjnie wpływać powinny) posłuży nam współczynnik korelacji Pearsona i test do badania jego istotności.

Sprawdźmy najpierw uśrednione wyniki wszystkich podskal BPRS:

```
data:  MANSA and bprs
t = -7.6558, df = 640, p-value = 7.113e-14
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.3589821 -0.2171346
sample estimates:
       cor
-0.2896479
```

BPRS — objawy maniakalne:

```
data:  MANSA and bprs.m
t = -2.5943, df = 640, p-value = 0.009695
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.17798960 -0.02483069
sample estimates:
       cor
-0.1020147
```

BPRS — objawy negatywne:

```
data:  MANSA and bprs.n
t = -2.9436, df = 640, p-value = 0.003362
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.19124632 -0.03854143
sample estimates:
       cor
-0.1155767
```

BPRS — objawy pozytywne:

```
data:  MANSA and bprs.p
t = -4.0444, df = 640, p-value = 5.885e-05
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.2324065 -0.0814809
sample estimates:
       cor
-0.1578655
```

BPRS — objawy depresyjne:

```
data:  MANSA and bprs.d
t = -9.3193, df = 640, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.4120279 -0.2756622
sample estimates:
       cor
-0.3456688
```

Widzimy że wszystkie te korelacje są istotne. Najbardziej z wynikiem MANSY koreluje poziom depresji, co jest wynikiem dość oczywistym.

Podsumowując, zmiennymi korelującymi z testem MANSA są:

- wszystkie wyniki testu BPRS,
- ośrodek, w którym leczy się pacjent (w Londynie i Wrocławiu niższe wyniki),
- fakt mieszkania samotnie\z kimś (niższa MANSA u osób, które mieszkają same).

3.2. Klasyfikacja z użyciem drzew decyzyjnych ze względu na wynik testu MANSA

Zmienną, ze względu na którą będę dokonywać klasyfikacji, jest MANSA. Muszę zatem nadać tej zmiennej charakter zmiennej typu czynnikowego. Zrobię to przez podzielenie skali testu MANSA na trzy przedziały.

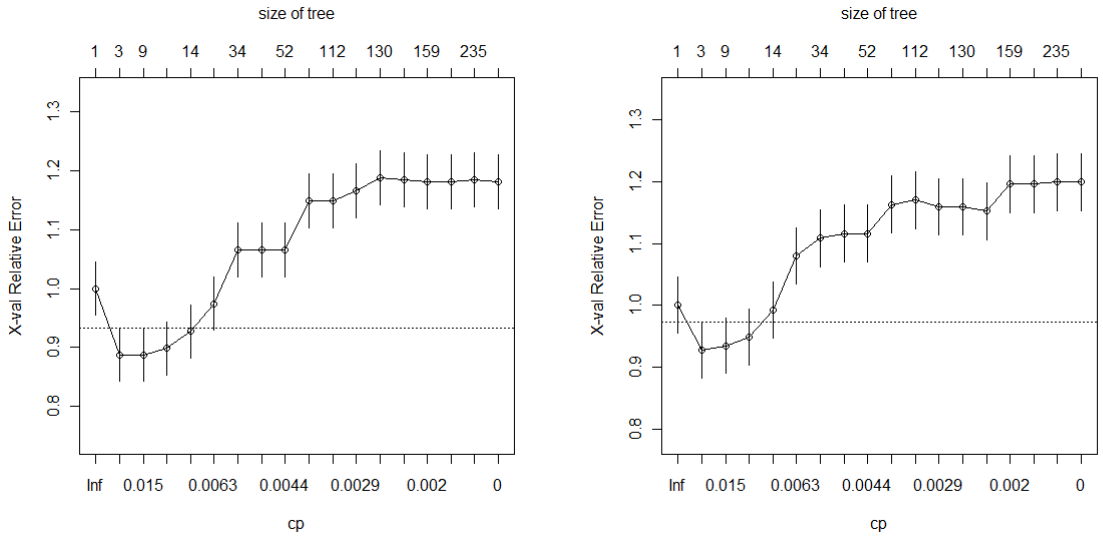
```
> MANSAd=cut(MANSA, 3)
> table(MANSAd)
MANSAd
(0.994,2.89]  (2.89,4.78]  (4.78,6.67]
          37           366           239
```

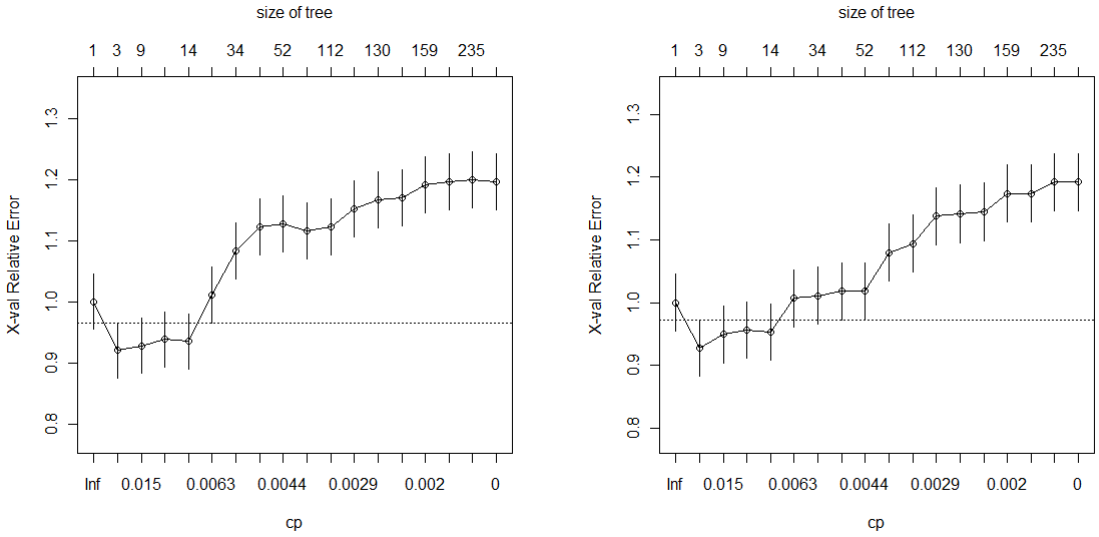
3.2.1. Trenowanie drzewa i analiza jego własności

Na początek zbudujmy drzewo pełne na podstawie wszystkich zmiennych, jakimi dysponujemy. Pozwoli nam to na przeanalizowanie zachowania drzewa w zależności od wszystkich tych zmiennych oraz cp.

```
m.tree.rpart <- rpart(MANSAd ~ Osrodek + tryb + cywilny + plec +
dzieci + mieszka + edukacja + praca + bprs + bprs.m + bprs.n +
bprs.p + bprs.d, method ="class", data=daneE, control =
rpart.control(cp=0.0,minsplit=0,minbucket=0))
```

Tym razem wybór opdowiedniego cp nie jest tak problematyczny, jak w poprzednim rozdziale. Przyjrzyjmy się pierwszym czterem wykresom, jakie dostaliśmy:





Rys.21. Wykresy cp dla drzewa pełnego uzyskanego z wykorzystaniem wszystkich zmiennych.

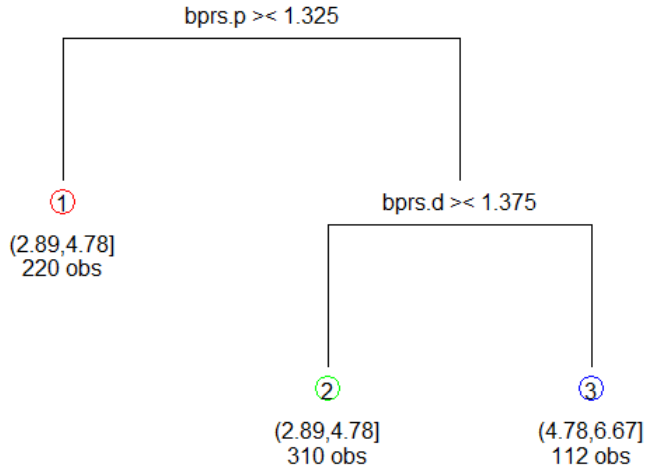
```
> printcp(m.tree.rpart)

Classification tree:
rpart(formula = MANSAd ~ Osrodek + tryb + cywilny + plec +
    dzieci + mieszka + edukacja + praca + bprs + bprs.m +
    bprs.n + bprs.p +     bprs.d, data = daneE, method = "class",
    control = rpart.control(cp =0, minsplit =0, minbucket =0))

Variables actually used in tree construction:
 [1] bprs     bprs.d   bprs.m   bprs.n   bprs.p   cywilny  dzieci
edukacja
 [9] mieszka  Osrodek  plec     praca    tryb

Root node error: 276/642 = 0.42991

n= 642

          CP nsplit rel error  xerror     xstd
1  0.0561594      0 1.0000000 1.00000 0.045448
2  0.0166667      2 0.8876812 0.93841 0.045037
3  0.0144928      8 0.7862319 0.94928 0.045120
4  0.0090580     10 0.7572464 0.97464 0.045295
5  0.0072464     13 0.7282609 0.98913 0.045386
6  0.0054348     17 0.6992754 1.03986 0.045643
7  0.0050725     38 0.5724638 1.05072 0.045687
8  0.0048309     43 0.5471014 1.05072 0.045687
9  0.0045290     49 0.5072464 1.07246 0.045762
10 0.0036232     66 0.4202899 1.07246 0.045762
11 0.0031056    124 0.2101449 1.13768 0.045891
12 0.0027174    132 0.1847826 1.13406 0.045887
```

```
13 0.0024155     140 0.1630435 1.12681 0.045879
14 0.0021739     153 0.1268116 1.12681 0.045879
15 0.0018116     161 0.1086957 1.14493 0.045896
16 0.0012077     219 0.0036232 1.14493 0.045896
17 0.0000000     222 0.0000000 1.13768 0.045891
```

3.2.2. Przycinanie drzewa i wynik klasyfikacji

Powyższe informacje stanowią podstawę do wyboru cp z przedziału $[0.0166667, 0.0561594)$. Ustalając cp=0.05, dostajemy drzewo o rozmiarze 3, a więc zgodnie z powższym to o najmniejszym błędzie klasyfikacji.

```
> mp.tree.rpart <-  prune.rpart(m.tree.rpart,cp=0.05)
n= 642

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

1) root 642 276 (2.89,4.78]
                (0.057632399 0.570093458 0.372274143)
  2) bprs.p>=1.325 220  70 (2.89,4.78]
                (0.104545455 0.681818182 0.213636364) *
  3) bprs.p< 1.325 422 206 (2.89,4.78]
                (0.033175355 0.511848341 0.454976303)
    6) bprs.d>=1.375 310 134 (2.89,4.78]
                (0.041935484 0.567741935 0.390322581) *
    7) bprs.d< 1.375 112  41 (4.78,6.67]
                (0.008928571 0.357142857 0.633928571) *
```



Rys.22. Wykres drzewa narysowany funkcją *draw.tree()*.

Widzimy tu, że do budowy tego drzewa użyte zostały jedynie zmienne bprs.d i bprs.p (dodam, że w rozdziale drugim nie używałam ich, korzystałam jedynie z łącznego wyniku kwestionariusza BPRS - zmienna bprs).

Przyjrzyjmy się, co się dzieje w liściach (tylko tę część odpowiedzi na polecenie *summary* umieszczam).

```
> summary(mp.tree.rpart)

...

          CP nsplit rel error    xerror       xstd
1 0.05615942      0 1.0000000 1.0000000 0.04544839
2 0.05000000      2 0.8876812 0.9384058 0.04503733

...

Node number 2: 220 observations
  predicted class=(2.89,4.78]  expected loss=0.3181818
    class counts:    23   150    47
   probabilities: 0.105 0.682 0.214

...

Node number 6: 310 observations
  predicted class=(2.89,4.78]  expected loss=0.4322581
    class counts:    13   176   121
```

```
    probabilities: 0.042 0.568 0.390

Node number 7: 112 observations
  predicted class=(4.78,6.67]  expected loss=0.3660714
    class counts:     1    40    71
   probabilities: 0.009 0.357 0.634
```

W węźle:

- nr 2 znaleźli się pacjenci, którzy mieli $\geqslant 1.325$ poziom objawów pozytywnych (przypomnę, że skala jest siedmiopunktowa) — zostali zakwalifikowani jako średnio zadowoleni z życia;
- nr 6 znalazły się osoby, które miały < 1.325 poziom objawów pozytywnych oraz których objawy depresyjne były $\geqslant 1.375$ — zostały zakwalifikowane jako średnio zadowolone;
- nr 7 znaleźli się natomiast badani, którzy mieli < 1.325 poziom objawów pozytywnych oraz którzy objawy depresyjne mieli < 1.375 — zostali oni zakwalifikowani, jako zadowoleni z życia.

```
> xpred.rpart(mp.tree.rpart, xval=10)
     0.52807971 0.05299029
1             2          3
2             2          2
3             2          2
4             2          2
5             2          2
6             2          2
7             2          2
8             2          2
9             2          2
10            2          2
11            2          2
12            2          2
13            2          2
14            2          2
15            2          2
16            2          2
17            2          2
18            2          3
19            2          2
20            2          2
...
```

Ze względu na dużą liczbę rekordów nie zamieszczam ich wszystkich. Pozwolę sobie jedynie zamieścić informację o liczbie osób, które mają określony poziom zadowolenia z życia zgodnie z kroswalidacyjnym klasyfikowaniem przez drzewo *mp.tree.rpart*.

```
> table(xpred.rpart(mp.tree.rpart,xval=15)[,1])

  2
642
```

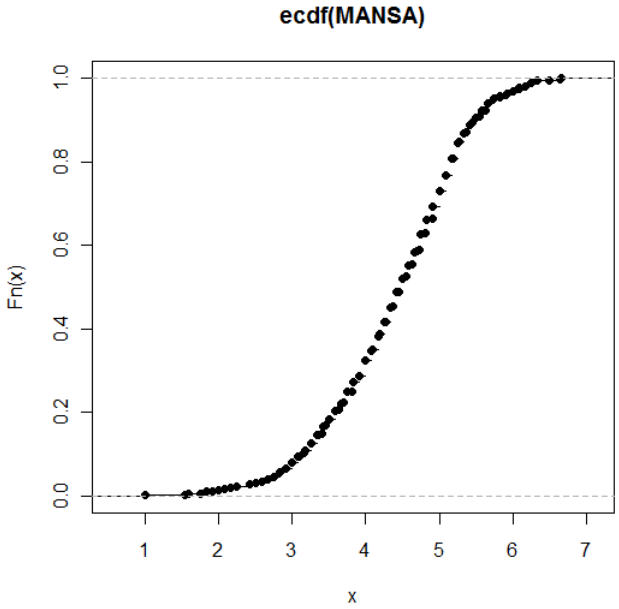
```
> table(xpred.rpart(mp.tree.rpart, xval=15)[,2])

  2   3
561  81
```

Warto sprawdzić, ile obserwacji faktycznie należy do której klasy.

```
(0.994,2.89]  (2.89,4.78]  (4.78,6.67]
          37           366           239
```

Pierwszy wynik został uzyskany dla cp= 0.52807971, natomiast drugi dla cp= 0.05299029.

Patrząc na rozkład wyników testu MANSA, taki efekt nie powinien dziwić. Spójrzmy:

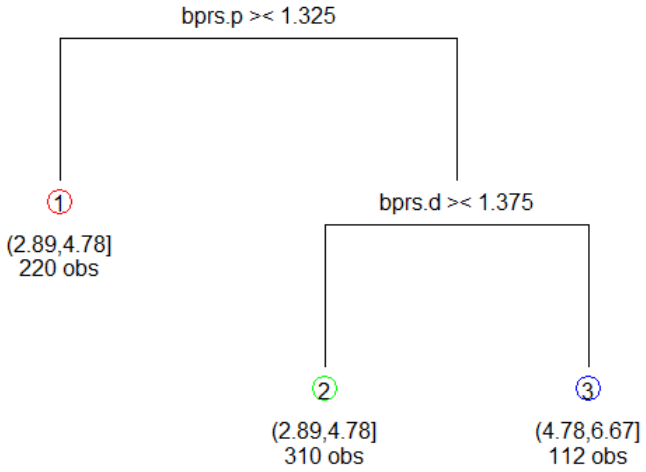


Rys.23. Dystrybuanta empiryczna zmiennej MANSA.

Widzimy, że mało jest obserwacji z niskim wynikiem testu MANSA, co zresztą potwierdza liczebność klas w próbie uczącej umieszczona powyżej.

Podusmowanie

Podsumowując, na podstawie zadanej próby uczącej usyskaliśmy następujące drzewo:



Rys.24. Wybrany klasyfikator.

Atrybutami decydującymi o podziałach są zmienne, które wykazały korelację z testem MANSA — objawy pozytywne oraz depresja. Nietrudno zauważyć, że nie używałam explicite algorytmów opisanych powyżej. Wynika to z zalety programu R, jaką są funkcje dostępne w tym programie. Warto mieć świadomość, jak wyglądają algorytmy przycinania, by wiedzieć, jak działają funkcje w R. Korzystanie z przedstawionych w tej pracy funkcji służących do przycinania znacząco ułatwia pracę z drzewami.

Dodatek A

Przygotowanie danych

Dane z programu EDEN, które posłużyły mi do zobrazowania działania drzew decyzyjnych, można znaleźć pod adresem: http://www.biecek.pl/MIMUW/index.php/SL2010/Opis

Natomiast, aby używać ich tak jak w pracy, nadałam im odpowiednie, wygodne w użyciu nazwy.

```
daneE=read.table("c:/katalog_z_danymi/EDEN.csv", sep=";", header=T)
attach(daneE)
bprs.m=daneE[,5]
bprs.n=daneE[,6]
bprs.p=daneE[,7]
bprs.d=daneE[,8]
bprs=daneE[,9]
dzieci=daneE[,12]
plec=daneE[,11]
mieszka=daneE[,13]
edukacja=daneE[,14]
praca=daneE[,15]
cywilny=daneE[,4]
tryb=daneE[,3]
```

Aby korzystać z opisanych w niniejszej pracy funkcji należy oczywiście załadować odpowiednie pakiety. Poniższe komendy ładują odpowiednie pakiety.

```
library(rpart)
library(tree)
library(party)
library(maptree)
```

Lista funkcji użytych w pracy:

- rpart, printcp, plotcp, prune.rpart, xpred.rpart z pakietu **rpart**;
- tree, prune.misclass z pakietu **tree**;
- ctree z pakietu **party**;
- group.tree, draw.tree z pakietu **maptree**;
- plot, text, summary, predict, cut, table, cor.test, chisq.test z innych pakietów.

Bibliografia

[KC] Koronacki J., Ćwik J., *Statystyczne systemy uczące się*, Wydawnictwa Naukowo-Techniczne, Warszawa 2005.

[B] Biecek P. *Przewodnik po pakiecie R*, Wrocław 2008

[WG] Walesiak M., Gatnar E., *Statystyczna analiza danych z wykorzystaniem programu R*, Wydawnictwo Naukowe PWN, Warszawa 2009.

[TA] Therneau T. M., Atkinson E. J., *An Instroduction to Recursive Partitioning Using the RPART Routines*, Mayo Foundation, September 3, 1997.

[EDEN] http://www.edenstudy.com/pl

[MANSA] http://en.wikipedia.org/wiki/MANSA

[BPRS] http://en.wikipedia.org/wiki/Brief_Psychiatric_Rating_Scale