# LABORATORIUM 12

- Drzewa klasyfikujące - wprowadzenie
- Reguły podziału
- Miary różnorodności
- Optymalizacja drzewa
- Pakiet rpart

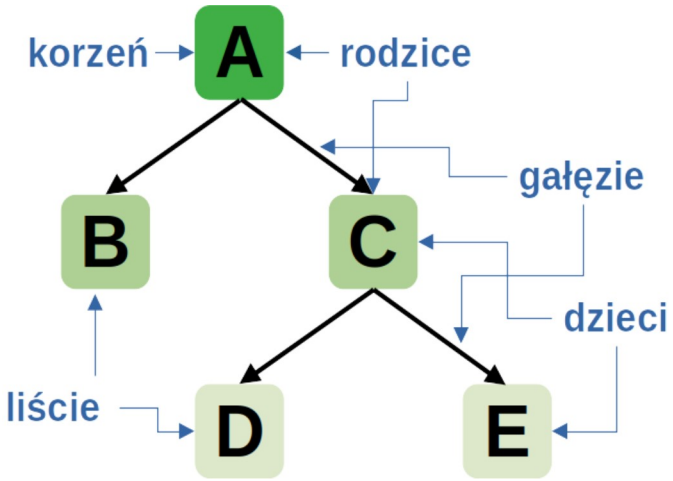
## Drzewa klasyfikujące - wprowadzenie

Drzewo klasyfikujące jest bardzo efektywnym algorytmem klasyfikacji pod nadzorem, na który składają się dwa elementy:

- struktura - drzewo skierowane, czyli acykliczny, spójny graf skierowany,
- reguły podziału - umieszczone w węzłach nie będących liśćmi, określają w jaki sposób ma zostać podzielona próba ucząca.

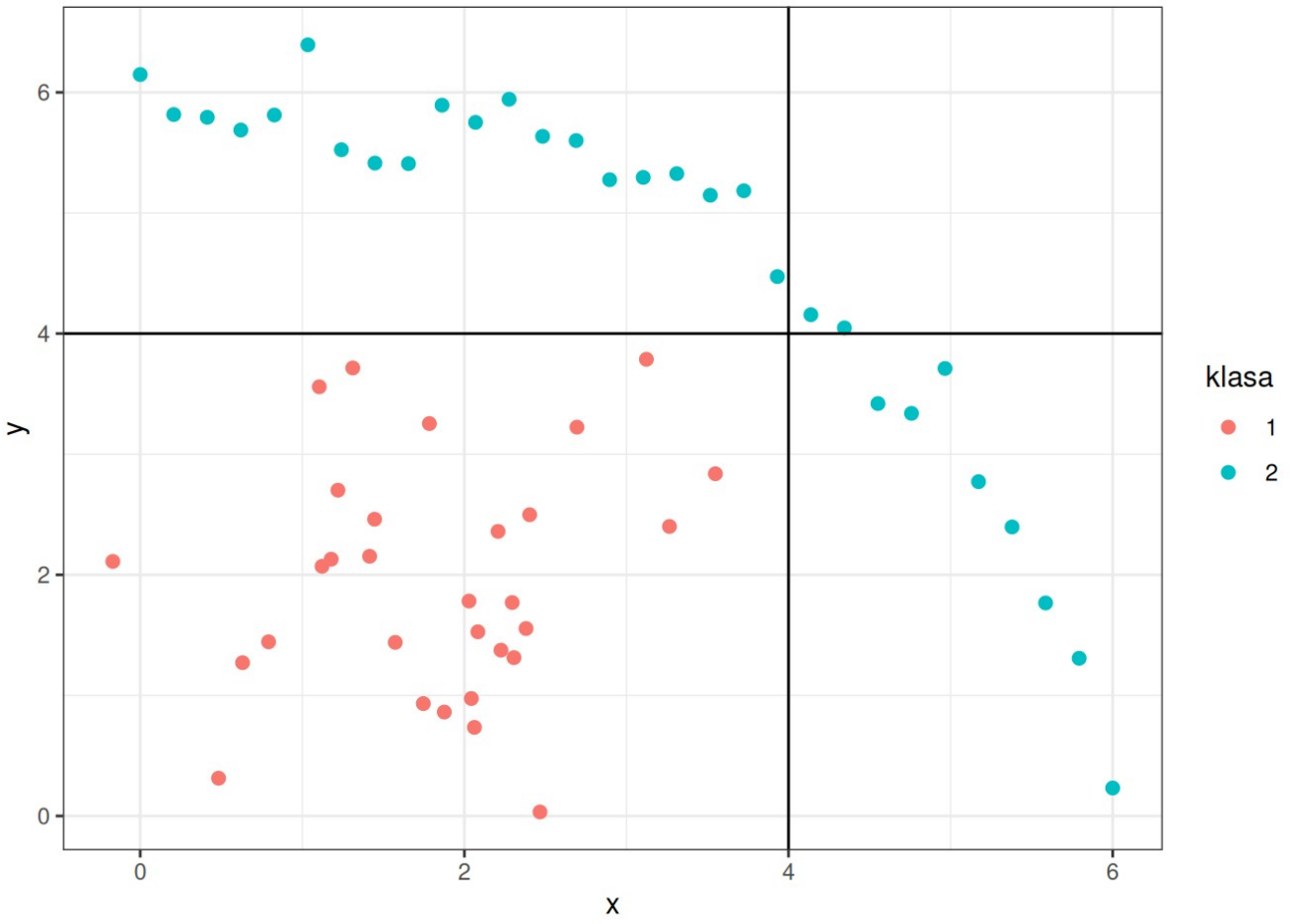
Poniżej przedstawiona jest przykładowa struktura drzewa. Wyróżniamy następujące elementy drzewa:

- węzły - elementy A-E na rysunku poniżej, w węzłach umieszczone są reguły podziału,
- gałęzie - krawędzie grafu skierowane od rodziców do dzieci,
- rodzice (A, B, C) - węzły, z których wychodzą gałęzie do innych węzłów (dzieci),
- dzieci (B-E) - węzły, do których wchodzą gałęzie z innych węzłów (rodziców),
- potomkowie - dzieci oraz dzieci ich dzieci,
- korzeń - węzeł bez rodzica, jeden w całym drzewie,
- liście - węzły bez dzieci.

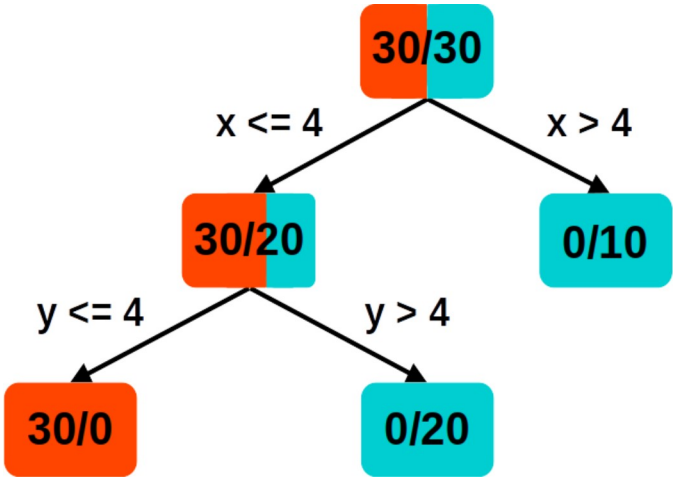



Według konwencji drzewa rysuje się rosnący od góry do dołu. Od korzenia do każdego liścia prowadzi tylko jedna ścieżka. W korzeniu jest skupiona cała próba ucząca (PU), której kolejne elementy są przesuwane w dół wzdłuż gałęzi. Decyzja, którymi gałęziami podążą elementy PU jest podejmowana w węzłach przy pomocy reguł podziału. Poniżej znajduje się prosty przykład danych, które można bardzo łatwo sklasyfikować przy pomocy drzewa z zaledwie pięcioma węzłami.

```
library(MASS)
library(ggplot2)
theme_set (theme_bw())
set.seed(123)

n <- 30
m1 <- c (2, 2)
S1 <- matrix (c (1,0,0,1), 2, 2)
x1 <- mvrnorm (n, m1, S1)

r <- 6
x2 <- seq (0, r, length.out = n)
y2 <- sqrt(r^2-x2^2)+runif(n,-0.5,0.5)

dane <- data.frame (x = c (x1[,1], x2),
                    y = c (x1[,2], y2),
                    klasa = as.factor (rep (c ("1","2"), each = n)))

ggplot (dane, aes (x=x, y=y)) +
  geom_point (aes (color=klasa), size = 2) +
  geom_vline (xintercept = 4) +
  geom_hline (yintercept = 4)
```




Na powyższym wykresie bardzo łatwo zauważyć proste reguły podziału ze względu na zmienne $x$ i $y$. Drzewo nauczone na podstawie takiej próby uczącej mogłoby wyglądać następująco:




## Reguły podziału

Podział podpróby uczącej dokonywany jest tylko na podstawie elementów PU w danym węźle i polega na najlepszym rozdzieleniu podpróby na dwie części przechodzące do węzłów dzieci. Najlepszy podział to taki, który maksymalizuje różnice pomiędzy otrzymanymi podzbiorami. Aby go znaleźć, potrzebne są:

- stosowna miara różnorodności klas w węźle,
- miara różnicy między różnorodnością klas w węźle-rodzicu i węzłach-dzieciach,
- algorytm maksymalizacji różnicy różnorodności.

Rozważmy problem klasyfikacji o klasach $k = 1, 2, 3, \ldots, g$. Niech $n(m)$ będzie liczebnością próby w węźle $m$, a $n_j(m)$ będzie liczbą obserwacji z klasy $j$ w węźle $m$. Wtedy ułamek obserwacji z klasy $j$ w węźle $m$ można zapisać jako $\hat{p}_j(m) = n_j(m)/n(m)$. Obserwacje w węźle $m$ są przyporządkowane do klasy $k$ spełniającej zależność

$$k(m) = \arg\max_{j=1,2,\ldots,g} \hat{p}_j(m).$$

Jeśli węzeł $m$ jest liściem, $k(m)$ jest ostatecznym wynikiem klasyfikacji obserwacji w węźle $m$, w przeciwnym razie $k(m)$ stanowi informację o tym, która klasa jest najliczniej reprezentowana w węźle.

## Miary różnorodności

Sensowna miara różnorodności to taka, która przyjmuje wartość 0 gdy wszystkie obserwacje należą do jednej klasy i wartość maksymalną, gdy mamy do czynienia z jednorodnym rozkładem klas w węźle, tzn. $\hat{p}_1(m) = \hat{p}_2(m) = \cdots = \hat{p}_g(m) = 1/g$. Przykładowymi miarami różnorodności są:

- ułamek błędnych klasyfikacji $Q_f(m) = 1 - \hat{p}_{k(m)}(m)$
- indeks Giniego $Q_g(m) = \sum_{j=1}^{g} \hat{p}_j(m)(1 - \hat{p}_j(m))$
- entropia $Q_e(m) = -\sum_{j=1}^{g} \hat{p}_j(m) \log \hat{p}_j(m)$

Rozpatrzmy przypadek, w którym węzły $m_L$ i $m_R$ są dziećmi węzła $m$, oraz $\hat{p}_L = n(m_L)/n(m)$ jest ułamkiem elementów PU, które z węzła $m$ przeszły do węzła $m_L$. Odpowiednio $\hat{p}_R = 1 - \hat{p}_L$ jest ułamkiem elementów PU, które z węzła $m$ przeszły do węzła $m_R$. Różnicę między różnorodnością klas węzła-rodzica i węzłów-dzieci definiuje się jako

$$\Delta Q(m) = Q(m) - Q(m_L, m_R),$$

gdzie $Q(m_L, m_R) = \hat{p}_L Q(m_L) + \hat{p}_L Q(m_R)$ jest łączną miarą różnorodności klas węzłów-dzieci węzła $m$. Generalnie przyjmuje się, że wskaźnik Giniego i entropia są bardziej czułe na zmiany różnorodności

klas w węzłach. Algorytm sprawdza po kolei wszystkie możliwe **podziały monotoniczne** (czyli postaci $x^{(l)} \leqslant c$, gdzie $c$ to jakaś zaobserwowana w PU wartość zmiennej) dla każdego atrybutu obserwacji i wybiera taki podział, który maksymalizuje $\Delta Q(m)$.

## Optymalizacja drzewa

Nawet posiadając algorytm tworzenia drzewa nie jest oczywiste kiedy zakończyć jego konstrukcję. Naiwnie można by sądzić, że należy kontynować budowanie drzewa do momentu, aż w liściach pozostaną tylko obserwacje z jednej klasy (czyli różnorodność klas w liściach będzie wynosiła zero). Jednak takie podejście doprowadzi do nadmiernego dopasowania drzewa do próby uczącej, czyli do **przetrenowania**. Najczęściej stosuje się etapową konstrukcję drzewa:

1. Rozbudowujemy drzewo tak długo jak to możliwe albo do spełnienia pewnej naturalnej reguły zatrzymania budowy, np.
    - w liściach są elementy tylko jednej klasy,
    - w liściach są obserwacje o tych samych wartościach, chociaż należące do różnych klas,
    - uznanie z góry na liść węzła, do którego dotarło nie więcej niż 5 elementów PU,
    - osiągnięto ustaloną, maksymalną odległość od korzenia do liścia.
2. Sprawdzamy zdolność klasyfikacyjną drzewa na próbie testowej lub walidacyjnej.
3. Przeprowadzamy przycinanie drzewa, tak aby otrzymać największą zdolność klasyfikacyjną.

Istnieją różne algorytmy przycinania drzewa, np. **algorytm kosztu-złożoności**, który poszukuje kompromisu pomiędzy kosztem dokonania błędnej klasyfikacji i kosztem wynikającym z konieczności zbudowania drzewa (proporcjonalnym do liczby liści). Celem algorytmu jest zminimalizować wielkość $R_\alpha(\mathcal{T}) = \rho(\mathcal{T}) + \alpha N_l(\mathcal{T})$, gdzie $\mathcal{T}$ oznacza dane drzewo, $\rho(\mathcal{T}) = 1 - ACC$ frakcję błędnych klasyfikacji na drzewie $\mathcal{T}$, $N_l(\mathcal{T})$ to liczba liści w tym drzewie, a symbolem $\alpha$ oznacza się współczynnik złożoności, który jest parametrem algorytmu.

## Pakiet rpart

W pakiecie R możliwe jest szybkie stworzenie drzewa klasyfikującego przy pomocy funkcji `rpart()` z biblioteki `rpart`. Parametr `minsplit` określa minimalną liczbę elementów, które muszą się znaleźć w weźle, aby dochodziło do podziału, natomiast przy pomocy parametru `minbucket` wybieramy minimalną liczbę obserwacji, która musi się znaleźć w węźle, aby mógł być uznany za liść. Pozostałe parametry to m.in. `cp` (współczynnik złożoności) i `maxdepth` (największa odległość między korzeniem a liśćmi).

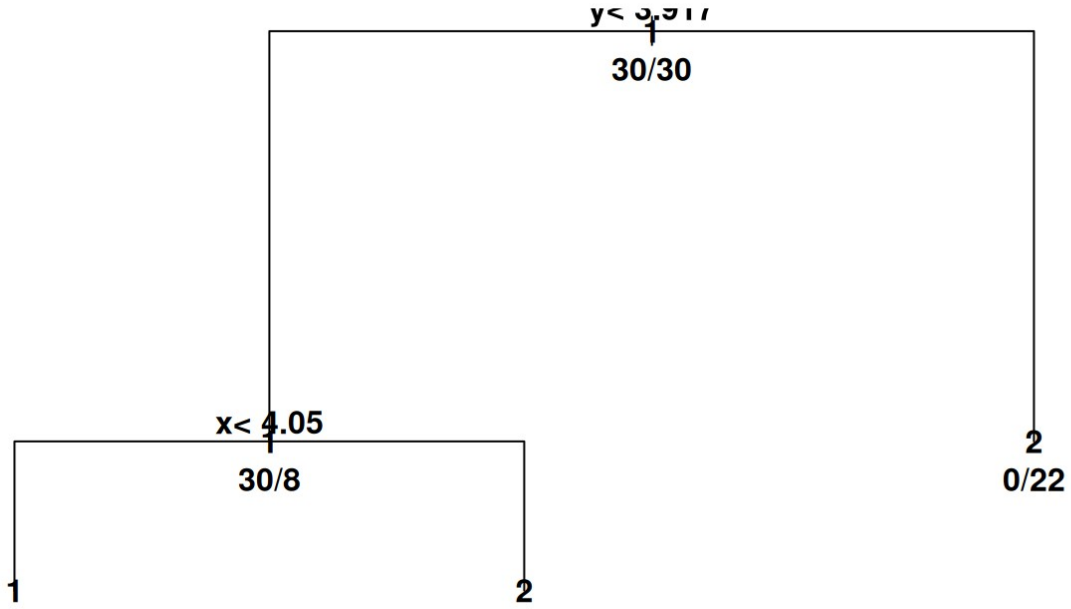
```
library (rpart)

# Uczenie drzewa
tree <- rpart(klasa ~ x + y, dane, minsplit = 2, minbucket = 1)
tree
```

```
## n= 60
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 60 30 1 (0.5000000 0.5000000)
##   2) y< 3.91716 38  8 1 (0.7894737 0.2105263)
##     4) x< 4.050238 30  0 1 (1.0000000 0.0000000) *
##     5) x>=4.050238 8  0 2 (0.0000000 1.0000000) *
##   3) y>=3.91716 22  0 2 (0.0000000 1.0000000) *
```

```
summary (tree)
```

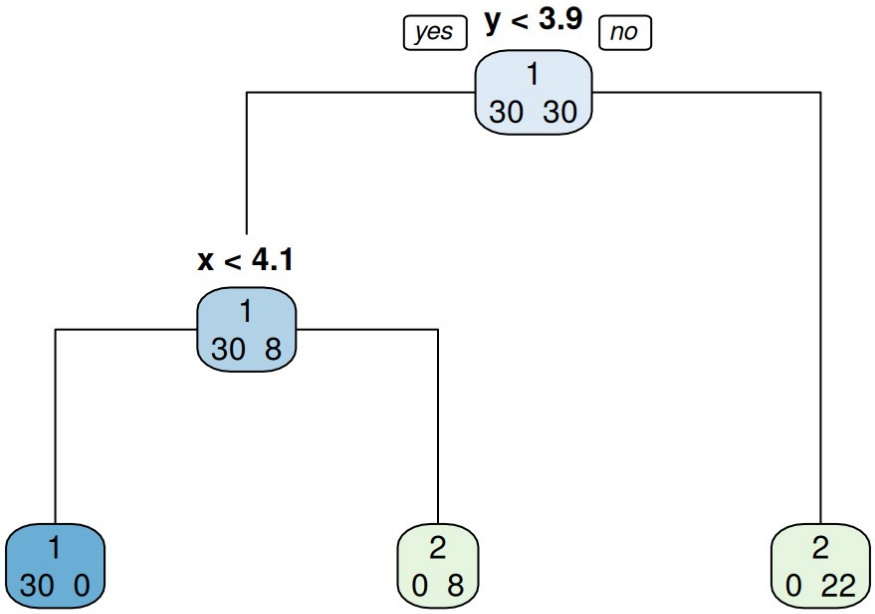
```
## Call:
## rpart(formula = klasa ~ x + y, data = dane, minsplit = 2, minbucket = 1)
##   n= 60
##
##          CP nsplit rel error    xerror      xstd
## 1 0.7333333      0 1.0000000 1.4000000 0.1183216
## 2 0.2666667      1 0.2666667 0.5000000 0.1118034
## 3 0.0100000      2 0.0000000 0.2333333 0.0828877
##
## Variable importance
##  y  x
## 55 45
##
## Node number 1: 60 observations,    complexity param=0.7333333
##   predicted class=1  expected loss=0.5  P(node) =1
##     class counts:    30    30
##    probabilities: 0.500 0.500
##   left son=2 (38 obs) right son=3 (22 obs)
##   Primary splits:
##       y < 3.91716   to the left,  improve=17.368420, (0 missing)
##       x < 3.287871  to the left,  improve= 7.511111, (0 missing)
##   Surrogate splits:
##       x < 0.4486612 to the right, agree=0.667, adj=0.091, (0 split)
##
## Node number 2: 38 observations,    complexity param=0.2666667
##   predicted class=1  expected loss=0.2105263  P(node) =0.6333333
##     class counts:    30     8
##    probabilities: 0.789 0.211
##   left son=4 (30 obs) right son=5 (8 obs)
##   Primary splits:
##       x < 4.050238  to the left,  improve=12.631580, (0 missing)
##       y < 3.296093  to the left,  improve= 1.194079, (0 missing)
##
## Node number 3: 22 observations
##   predicted class=2  expected loss=0  P(node) =0.3666667
##     class counts:     0    22
##    probabilities: 0.000 1.000
##
## Node number 4: 30 observations
##   predicted class=1  expected loss=0  P(node) =0.5
##     class counts:    30     0
##    probabilities: 1.000 0.000
##
## Node number 5: 8 observations
##   predicted class=2  expected loss=0  P(node) =0.1333333
##     class counts:     0     8
##    probabilities: 0.000 1.000
```

```
# Rysowanie drzewa
plot (tree)
text (tree, use.n = TRUE, all = TRUE, font = 2)
```



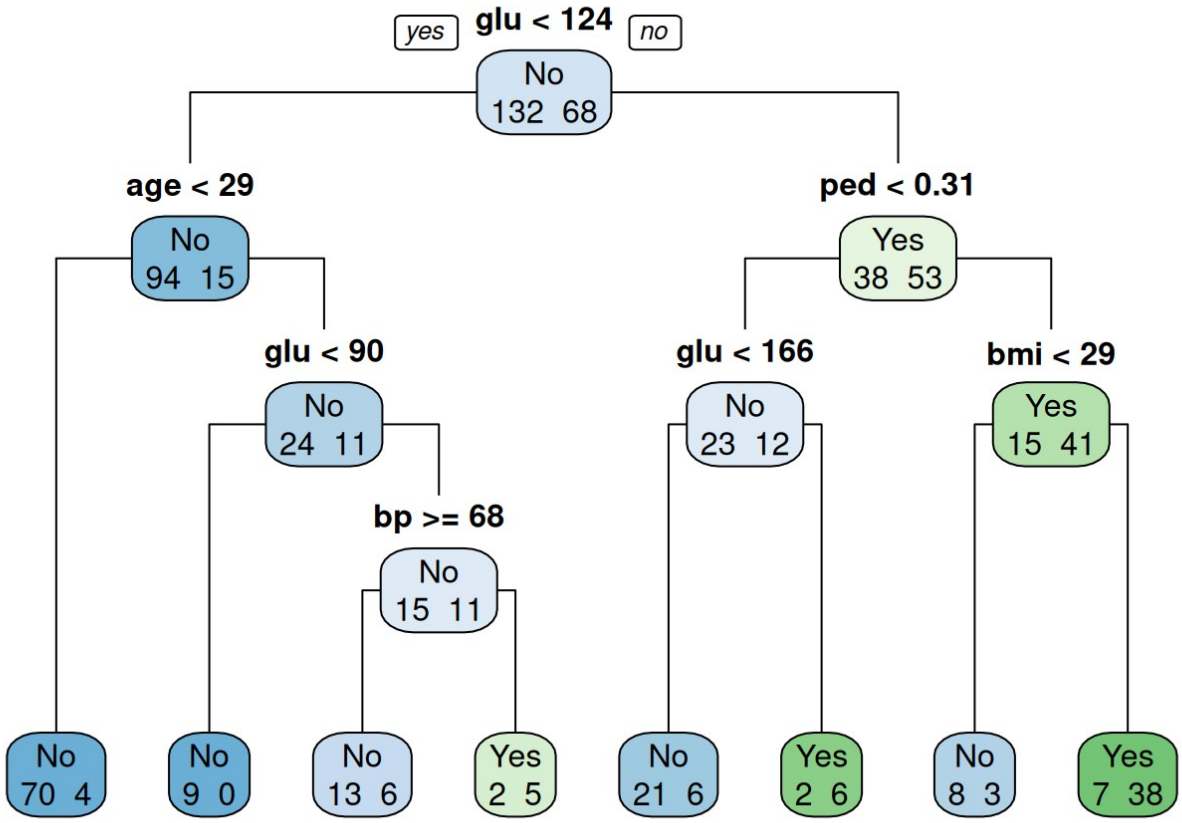

Otrzymany rysunek pozostawia wiele do życzenia. Aby go poprawić, użyjemy funkci `rpart.plot()` z biblioteki `rpart.plot` .

```
library(rpart.plot)
rpart.plot (tree, type = 1, extra = 1)
```




Przetestujmy drzewo klasyfikujące na bardziej wymagających danych, np. zbiorze danym `Pima` z biblioteki `MASS` .

```
library(MASS)
library(caret)

pima.tree <- rpart (type ~., Pima.tr)
rpart.plot (pima.tree, type = 1, extra = 1)
```

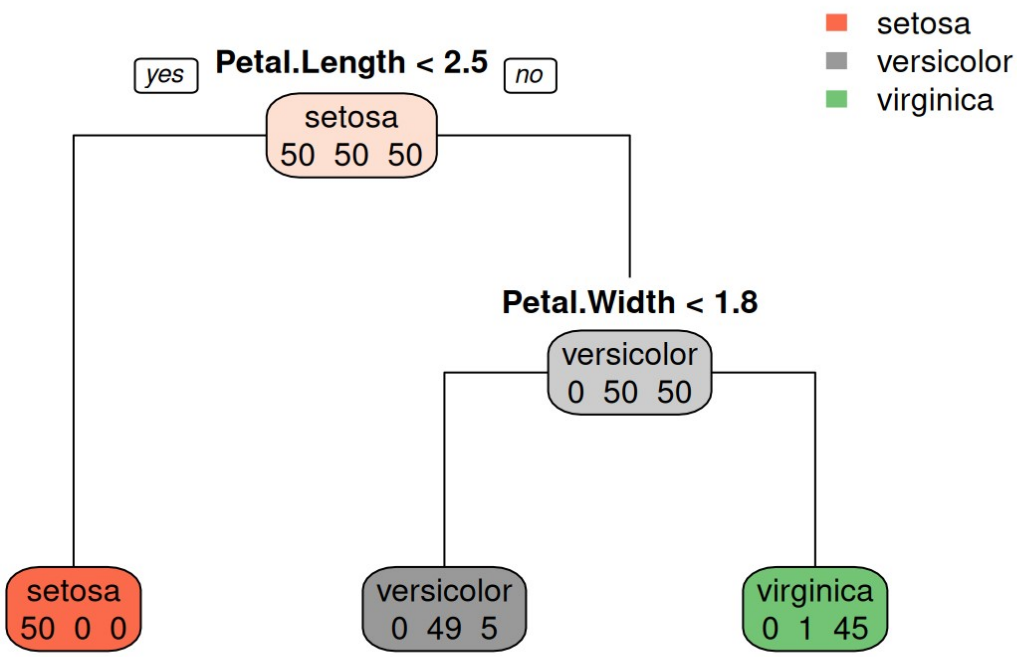



```
pima.predict <- predict (pima.tree, newdata = Pima.te, type = "class")
pima.true <- Pima.te[,8]

pima.cm <- confusionMatrix (pima.predict, pima.true, positive = "Yes")
pima.cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  182  48
##        Yes  41  61
##
##                Accuracy : 0.7319
##                  95% CI : (0.6808, 0.7788)
##     No Information Rate : 0.6717
##     P-Value [Acc > NIR] : 0.01042
##
##                   Kappa : 0.382
##
##  Mcnemar's Test P-Value : 0.52478
##
##             Sensitivity : 0.5596
##             Specificity : 0.8161
##          Pos Pred Value : 0.5980
##          Neg Pred Value : 0.7913
##              Prevalence : 0.3283
##          Detection Rate : 0.1837
##    Detection Prevalence : 0.3072
##       Balanced Accuracy : 0.6879
##
##        'Positive' Class : Yes
##
```

Rzecz jasna, nic nie stoi na przeszkodzie, aby w zbiorze danych występowały więcej niż dwie klasy. Poniżej przykład zastosowania drzewa do zbioru `iris`.

```
iris.tree <- rpart (Species ~., iris)
rpart.plot (iris.tree, type = 1, extra = 1)
```


setosa
versicolor
virginica
yes
Petal.Length < 2.5
no
setosa
50 50 50
Petal.Width < 1.8
versicolor
0 50 50
setosa
50 0 0
versicolor
0 49 5
virginica
0 1 45