[連載]　フリーソフトによるデータ解析・マイニング　第19回

ツリーモデル(木)

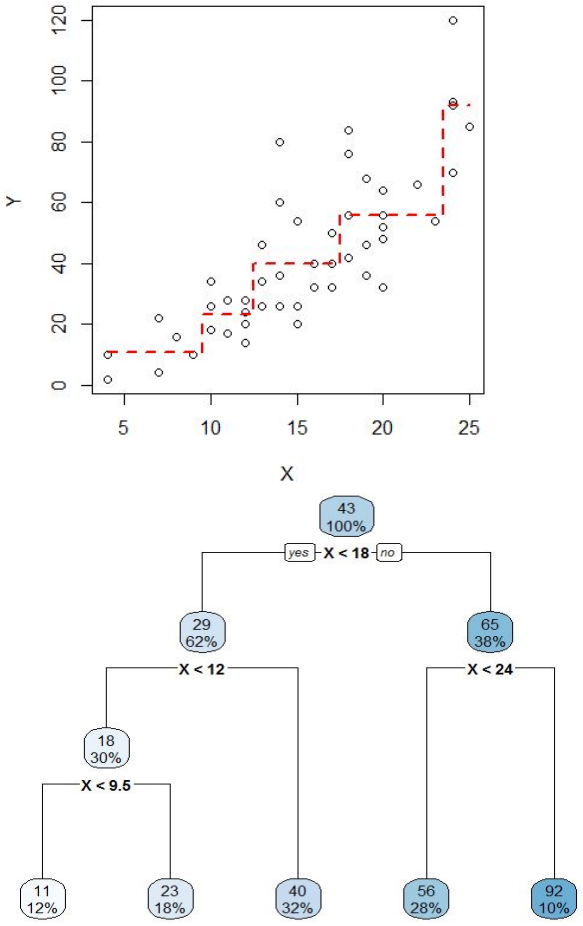
一般化線形モデルや混合効果モデルなどはデータが何らかの確率分布にしたがうことを前提としてモデルを構築している。しかし、確率分布の仮定は問題によっては条件が厳しい場合もある。確率分布を意識せずにデータの回帰、あるいは判別のモデルを構築する方法としてツリーモデルがある。本章ではツリーモデルの概念、CART のアルゴリズム、CART の操作、いくつかの図示関数、木の生長とコントロール、木の剪定、回帰木などについて説明する。

1. ツリーモデルとは

ツリーモデル (tree-based model) は回帰分析、判別分析を行う方法であり、回帰の問題では回帰木 (regression tree)、判別・分類の問題では分類木 (classification tree) あるいは決定木 (decision tree) とよばれている。

ツリーモデルは、説明変数の値を分岐させ、それらを組み合わせて、予測・判別のモデルを構築する。分析の結果は「もし…であれば…」(IF-THEN)のような簡潔なルールを返し、またそのルールを木構造で図示することができるため、理解しやすい。

説明変数 x で被説明変数 y を説明する単回帰分析では関数式 $y = a + bx$ で当てはめた。回帰木では図1(a) のように階段型の折れ線で当てはめる。その結果は図1(b) のように逆さにした木の形で表現することができる。



(a)　回帰直線と回帰曲線　　(b)　樹木モデルによる回帰折れ線

図1　回帰木

sss

また、この結果は次のようなルールで表現することができる。ルールから初めの分岐は説明変数17.5から始まっていることがわかる。

```
Model formula: Y ~ X

[1] root
|   [2] X < 17.5
|   |   [3] X < 12.5
|   |   |   [4] X < 9.5: 10.667 (n = 6, err = 277.3)
|   |   |   [5] X >= 9.5: 23.222 (n = 9, err = 331.6)
|   |   [6] X >= 12.5: 39.750 (n = 16, err = 3535.0)
|   [7] X >= 17.5
|   |   [8] X < 23.5: 55.714 (n = 14, err = 2846.9)
|   |   [9] X >= 23.5: 92.000 (n = 5, err = 1318.0)
```

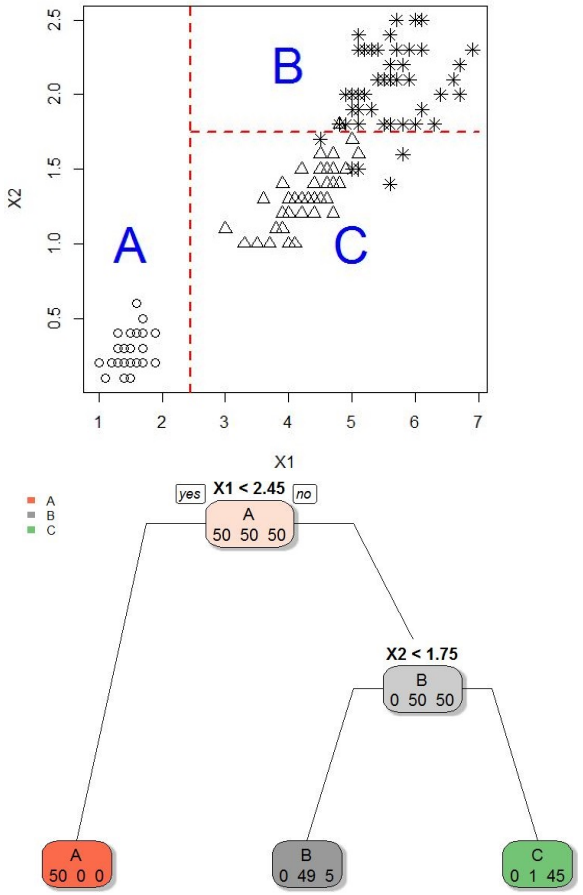
説明変数が17.5より大きい部分は、さらに2つの部分に分割している。説明変数が17.5から23.5までの y は55.714であり、説明変数が23.5以上のときの y は92.0である。この結果は次のように表現できる。

もし　$17.5 \leq x < 23.5$　であれば　$y = 55.714$
もし　$23.5 \leq x < 23.5$　であれば　$y = 92.0$

このようなルールを簡潔に書いたのが上記の縦破線付いている出力形式である。次の部分がこれに相当する。同じく説明変数が17.5より小さい部分もこのように表現することができる。

```
[7] X >= 17.5
|   |   [8] X < 23.5: 55.714 (n = 14, err = 2846.9)
|   |   [9] X >= 23.5: 92.000 (n = 5, err = 1318.0)
```

判別・分類を行う分類木は図2のように表現できる。図2(a)のように、分類木は説明変数と並行する直線で分割する。



(a) 分類木の分割図　　(b) 分類木

図2 分類木と分割図

回帰木の場合と同じく、分類木でも次のようなルールで表現できる。

```
Model formula:
lab ~ X1 + X2

[1] root |   [2] X1 < 2.45: A (n = 50, err = 0.0%)
|   [3] X1 >= 2.45
|   |   [4] X2 < 1.75: B (n = 54, err = 9.3%)
|   |   [5] X2 >= 1.75: C (n = 46, err = 2.2%)
```

このようにツリーモデルは、図3のような木の構造で図示化でき、IF-THENルールで表現する。根(ルート)、節(ノード)をまとめてノードとよぶ場合もある。本章でも特別説明がない限りノードとしたときにはルートを含むことにする。

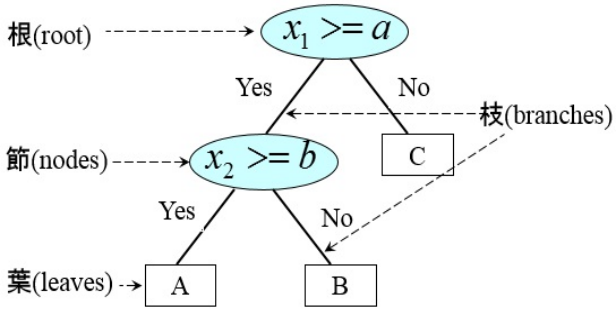


図3 ツリーモデルの視覚化

IF $x_1 < a$ THEN C
IF $x_1 >= a$ and $x_2 >= b$ THEN A
IF $x_1 >= a$ and $x_2 < b$ THEN B

ツリーモデルでは、枝を増やすことで複雑な分類問題や回帰問題に適応させることが可能である。ツリーモデルは結果と変数との関係が考察できるので便利である。また、高度な数学知識がなくても、その原理やアルゴリズムを理解することができるため、データマイニングの方法として広く用いるようになっている。

ツリーモデルは、説明変数を何らかの統計量に基づいて分割 (あるいは分岐) し、ルールを構築する。分割の基準となる統計量によって生成したルールが異なる。ツリーモデルに関する研究は、1960年代初期までさかのぼる。現段階で広く用いられているのはCART、CHAID、 C4.5/C5.0をベースとしたアルゴリズムである。

CART (classification and regression trees) は、カリフォルニア大学の L.Breiman、R.A. Olshen、C.J.Stone、スタンフォード大学の J.H.Friedman が1970年代初めごろから共同研究を始め、1980年代初めごろに公開したアルゴリズムである。CARTは、説明変数を2進分岐させ、2進木を生成する。分岐の評価基準として経済学者ジニ (Gini) によって提案されているジニ係数のアイディアに基づいたをジニ分散指標 (Gini's diversity index) と情報利得 (information gain) が用いられている。

CHAID (Chi-squared automatic interaction detection) は J. A. Hartigan によって1975年に発表された。上記の3種類の中で最も古いアルゴリズムである。CHAIDは1963年J. A. Morgan らによって提案されたAID (automatic interaction detection) を発展させたもので、変数を分割する統計量としてカイ2乗統計量やF検定統計量を用いている。

C4.5/C5.0は、オーストラリアの J. Ross Quinlan が機械学習のアプローチで1986年に発表したID3 (iterative dichotomiser 3) を改良・発展させたものである。名前に用いた数値4.5、5.0はバージョンの記号である。C4.5/C5.0は、2進木に限らないのが CART との大きな違いである。C4.5/C5.0の前身である ID3 では、分割の評価基準として情報利得 (information gain) を用いたが、C4.5/C5.0では利得比 (gain ratio) を用いている。

これらのツリーモデルのアルゴリズムはそれぞれ特徴を持っている。これらのツリーモデルの大きい違いは木の生成・生長のために分岐の基準となる統計量以外に、木の剪定のアルゴリズムである。

木の生成・生長とは、データセットから木の幹・枝となる説明変数を選定し、分割基準に基づいて分割させ、木を生長させる。木の剪定とは、生長し過ぎた木を何らかの基準に基づいて枝刈りし、学習データにおける過剰に適合 (oveerfitting: オーバーフィット) しない簡潔なツリーモデルを構築する作業である。

2. 分割基準

CARTを例として枝の分岐のための変数の分割基準のアルゴリズムを説明する。CARTでは分割の基準となる統計量はジニ分散指標とシャノンのエントロピーを用いる。CART では、ジニ分散指標に基づいたジニ多様性指標とよぶ不純度 (Impurity) を用いる。不純度は、変数の分割前後の差分を用いる。まずジニ分散指標の定義式を次に示す。式の中の p_{tk} は、ノード t 内のカテゴリ k が正しく分類されている比率である。

$$G(t) = 1 - \sum_k p_{tk}^2$$

ジニ分散指標を用いた不純度は次のように計算する。$G(t_L)$、$G(t_R)$ は、それぞれノード t の左側と右側の枝のジニ分散指標である。p_t、p_L、p_R は、それぞれ分割する前と分割後の左側、右側の個体の比率である。

$$\Delta G(t) = p_t G(t) - p_L G(t_L) - p_R G(t_R)$$

CART では、G_t の代わりに情報量エントロピー

$$E(t) = -\sum_k p_{tk} \log(p_{tk})$$

$E(t)$ を用いることも可能である。式の中の p_{tk} はジニ分散指標の場合と同じである。

例を用いて説明するため、表1に天気 (風、気温) と海水浴に行くか、行かないかに関するデータを示す。ここでは風は「強」「弱」2つのカテゴリ、気温は「高」「中」「低」三つのカテゴリで記録されている。また海水浴に関しては「行く」「行かない」2つのカテゴリである。

表1　海水浴のデータ

風	気温	海水浴
弱	高	行く
弱	低	行かない
弱	高	行く
弱	中	行く
弱	低	行かない
弱	高	行かない
強	高	行かない
強	低	行かない
強	中	行かない
強	高	行く

海水浴に行く行動と風の要因の関連性を考察するため、両項目のクロス表を表2に示す。

表2　風と海水浴のクロス表

風/海水浴	行かない	行く
強	3	1
弱	3	3

X-squared = 0.0174, df = 1, p-value = 0.8952

海水浴に「行く」 (10件中4件) と「行かない」 (10件中6件) のジニ分散指標は

$$G(\text{海水浴}) = 1 - \left[\left(\frac{4}{10}\right)^2 + \left(\frac{6}{10}\right)^2\right] = 0.48$$

となる。ここで風の「強」「弱」のジニ分散指標は

$$G(\text{風} = \text{強}) = 1 - \left[\left(\frac{3}{4}\right)^2 + \left(\frac{1}{4}\right)^2\right] = 0.375$$

$$G(\text{風} = \text{弱}) = 1 - \left[\left(\frac{3}{6}\right)^2 + \left(\frac{3}{6}\right)^2\right] = 0.5$$

となる。このデータに基づいた不純度は次となる。

$$\begin{aligned}\Delta G(\text{風}) &= G(\text{海水浴}) - p(\text{強})G(\text{風} = \text{強}) - p(\text{弱})G(\text{風} = \text{弱}) \\ &= 0.48 - \frac{4}{10} \times 0.375 - \frac{6}{4} \times 0.5 = 0.3\end{aligned}$$

後に用いるので、ついでに表2のクロス表のカイ二乗検定の統計量を表の中に示しておく。

同様に気温についてもジニ分散指標と不純度を求めることができる。ただし、気温は三つのカテゴリであるので、「高」と「中・低」、「中」と「高・低」、「低」と「中・高」に分けて分割表を作成計算することが必要である。計算に用いるクロス表およびジニ分散指標、不純度を表3にまとめて示す。

表3　気温と海水浴との不純度

	行かない	行く
高	2	3
中・低	4	1

$$p(\text{高})G(\text{高}) = \frac{5}{10}\left\{1 - \left[\left(\frac{2}{5}\right)^2 + \left(\frac{3}{5}\right)^2\right]\right\} = 0.24$$

$$p(\text{中・低})G(\text{中・低}) = \frac{5}{10}\left\{1 - \left[\left(\frac{4}{5}\right)^2 + \left(\frac{1}{5}\right)^2\right]\right\} = 0.16$$

$$\Delta G(\text{高}|\text{中・低}) = 0.48 - 0.24 - 0.16 = 0.08$$

X-squared = 0.4167, df = 1, p-value = 0.5186			
	行かない	行く	$p(\text{中})G(\text{中}) = \frac{2}{10}\left\{1 - \left[\left(\frac{1}{2}\right)^2 + \left(\frac{1}{2}\right)^2\right]\right\} = 0.1$
中	1	1	$p(\text{高・低})G(\text{高・低}) = \frac{8}{10}\left\{1 - \left[\left(\frac{5}{8}\right)^2 + \left(\frac{3}{8}\right)^2\right]\right\} = 0.375$
高・低	5	3	$\Delta G(\text{中}\|\text{高・低}) = 0.48 - 0.1 - 0.375 = 0.005$
X-squared = 0.5, df = 1, p-value = 0.4795			
	行かない	行く	$p(\text{中・高})G(\text{中・高}) = \frac{7}{10}\left\{1 - \left[\left(\frac{3}{7}\right)^2 + \left(\frac{4}{7}\right)^2\right]\right\} = 0.343$
中・高	3	4	$p(\text{低})G(\text{低}) = \frac{3}{10}\left\{1 - \left[\left(\frac{3}{3}\right)^2 + \left(\frac{0}{3}\right)^2\right]\right\} = 0$
低	3	0	$\Delta G(\text{中・高}\|\text{低}) = 0.48 - 0.343 - 0 = 0.137$
X-squared = 0.5, df = 1, p-value = 0.4795			

CART ではこのように計算した不純度の中、値がもっとも大きいものを第1候補とする。よって、ここでは max(0.03, 0.08, 0.05, 0.137) = 0.137 であるので気温データを「中・高」と「低」に分ける候補を第1分岐点とする。

ついでに、カイ二乗値について比較すると表側が「中・高」「低」のケースのカイ二乗値 X-squared = 0.9722 が最も大きく、P値が最も小さい。ツリーモデルのアルゴリズム CHAID では、カイ二乗値を用いる。CART の不純度に基づいて作成したツリーモデルを図4に示す。

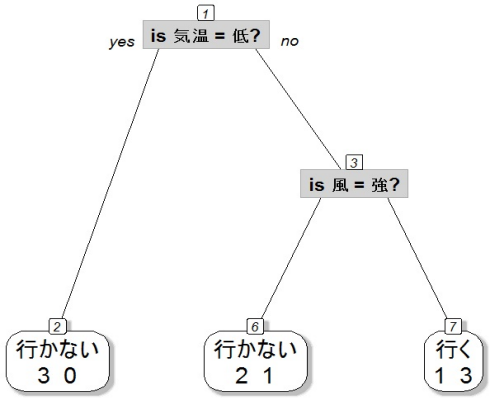


図4　データ表1の分類木

3. R上でのツリーモデルの操作

R には CART に関連するパッケージ tree、rpart、party、C4.5 に関するパッケージ RWeka、C5.0 などがある。本稿では rpart を用いることにする。パッケージ rpart は R をインストール際にインストールされている。パッケージ rpart(recursive partitioning and regression trees) で木を生成する関数は rpart である。

3.1　関数 rpart による木の生成

関数 rpart の書式を次に示す。ただし、引数は必要最小限のものを示している。より多くの引数やデフォルトの設定などは help(rpart) で確認することができる。

rpart (formula, data, method, control...)

formula は目的変数、説明変数を記述する書式であり、関数 lm の基本書式と同じである。引数 method は表4の中から1つを選択することができる。引数 method が指定されていない場合は、formula に指定されている被説明変数のデータ型 (量的データ、カウントデータ、カテゴリカルデータ、生存データ) から判断する。関数 rpart デフォルトではジニ分散指標による不純度を計算するよ

うに指定されている。情報量エントロピーを用いる際には、引数 split = “information”を用いる。引数 control には、木の生長をコントロールする条件を指定する。初期値は rpart.control に記録されている。

表4　引数 method のオプション

応答変数 y	引数の書き式
y が一般の量的変数	method = "anova"
y がポアソン分布	method = "poisson"
y が質的変数	method = "class"
y が生存データ	method = "exp"

例1　表1のデータを関数 rpart で分類木を生成せよ。

まず表1のデータを次のように R に入力する。

```
> 風<-c (rep("弱",6),rep("強",4))
> 気温<-c ("高","低","高","中","低","高","高","低","中","高")
> 海水浴<-c ("行く","行かない","行く","行く","行かない","行かない","行かない","行かない","行かない","行く")
> dat<-data.frame(風,気温,海水浴)
```

次のようにパッケージ rpart を読み込み、関数 rpart を実行する。関数 rpart では各枝における最小個体数のデフォルト値は minsplit = 20、葉の中の個体数の最小数は minbucket = round(minsplit/3) になっている。関数 rpart.control に記録されている。

用いたデータの個体数はわずか10であるので、minsplit のデフォルトのまま実行すると木が生成されない。そこで、ここでは minsplit = 3 にしる。必ず3にする必要はないが、一つの葉に1個体を許すのであれば3で十分である。個体数が20を大きく超えるときには、このように引数のオプションを調整しなくても木が生成される。

```
> library(rpart)
> res<-rpart(海水浴~.,dat, minsplit=3)
> res
n= 10

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

1) root 10 4 行かない (0.6000000 0.4000000)
  2) 気温=低 3  0 行かない (1.0000000 0.0000000) *
  3) 気温=高,中 7  3 行く (0.4285714 0.5714286)
    6) 風=強 3  1 行かない (0.6666667 0.3333333) *
    7) 風=弱 4  1 行く (0.2500000 0.7500000) *
```

木はまず「気温＝低」と「気温＝高・中」で分割している。「気温＝低」の枝には個体が3つ該当しているが、そのすべてが「いかない」ケースである。丸括弧内は各ケースに該当する相対比率である。「気温＝高・中」の枝には、個体が7つあり、その中「行かない」ケースが3つ、「行く」ケースが4つである。丸括弧内の数値はこの枝内での相対頻度である。

R には樹形モデルに関するパッケージが複数ある。パッケージ partykit の中の関数を用いると次のように出力形式を変えることができる。ルールの読み方を簡単に説明する。たとえば、枝[4] 《風 in 強: 行かない (n = 3, err = 33.3%) 》は、「行かない」に対する判別結果であり、個体数は3である。err = 33.3% は、3つの中1つが誤判別され、その比率 1/3=0.33 を百分率置き換えたものである。

```
> install.packages("partykit")
> library(partykit)
```

```
> res3<-as.party(res)
> print(res3)

Model formula:
海水浴 ~ 風 + 気温

Fitted party:
[1] root
|   [2] 気温 in 低: 行かない (n = 3, err = 0.0%)
|   [3] 気温 in 高, 中
|   |   [4] 風 in 強: 行かない (n = 3, err = 33.3%)
|   |   [5] 風 in 弱: 行く (n = 4, err = 25.0%)

Number of inner nodes: 2
Number of terminal nodes: 3
```

3.2 木の図示

(1) 関数 plot.rpart と text.rpart による図示

パッケージ rpart では、関数 plot.rpart と text.rpart を用いて生成したルールを図示することができる。

```
plot(x, uniform = FALSE, branch = 1, compress = FALSE, nspace, margin = 0, minbranch = 0.3, ...)
```

関数 plot.rpart の引数 x には関数 rpart の結果、引数 uniform はノード間の枝の長さに関する引数である。デフォルトは uniform = FALSE で、枝の長は分類のエラーの数に比例するように作成する。引数を uniform = TRUL にすると、ノード間の枝の長さは同じ長さとなる。

引数 branch は枝の角度を調整する。指定する値は0から1までの値を用いる。0の場合の角度が最も大きく、1の場合は垂直となる。デフォルトは branch = 1 になっている。

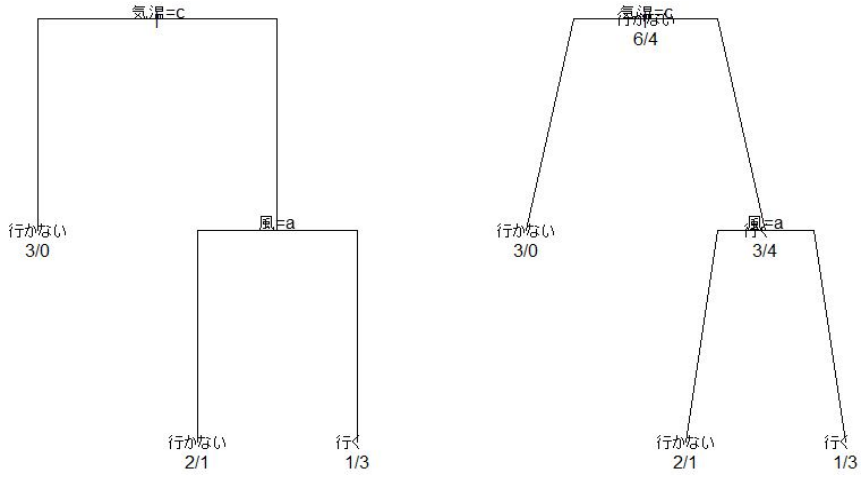
引数 margin は図の外枠の余白(マージン)を調整する。引数 margin の値が大きいほど図が小さくなり、余白が大きい。デフォルトは0.01になっているが、そのまま作成するとラベルが切れて見えない場合がある。

木のノードや葉のラベルを装飾する関数は text.rpart である。引数 use.n は、各ノードに含まれる個体の数の表示を指定する。デフォルトには use.n = FALSE になっている。引数が use.n = TRUE である場合は、ノードに含まれる数を表示する。

例2 例1で作成した分類木のオブジェクト res を用いて関数 plot で分類気を作成せよ。

枝の角度やラベル表示を変えた2つのグラフを次のように作成する。

```
> par(mfrow=c(1,2))
> plot(res,margin = 0.05)                    #  図(a)の作成
> text(res,use.n =TRUE)                      #  作成した木にラベルを付ける
> plot(res, margin = 0.05,branch=0.6)        #  図(b)の作成
> text(res, use.n =TRUE, all =TRUE)
```



(a) (b)

図5　分類木の図示

図5のようにパッケージ rpart の中の関数 plot.rpart と text.rpart を用いて作成した木のグラフは非常にシンプルであり、かつ文字列が線と重なる場合もある。そこで関数 rpart で生成したルールを図示する専用パッケージ rpart.plot が公開されている。

(2)　関数 rpart.plot を用いた図示

パッケージ rpart.plot の中にはグラフを作成する関数 rpart.plot と prp がある。関数 rpart.plot には多くの引数が用意されている。引数をすべて説明する紙面がないので、主な引数とオプションを表5に示す。

```
rpart.plot(x=stop("no 'x' arg"),type=0, extra=0, under=FALSE, clip.right.labs=TRUE,
fallen.leaves=FALSE, branch=if(fallen.leaves) 1 else .2, uniform=TRUE, digits=2, varlen=-8,
faclen=3,cex=NULL, tweak=1, compress=TRUE, ycompress=uniform, snip=FALSE,...)
```

表5　関数 rpart の 引数とオプション

引数とデフォルト値	オプションと機能
type=0	ノードラベルと分割情報のラベルの表示をコントロールする。 0~4の値を用いる。 0: ノードに分割ラベルを付ける。 1: 分割ラベルを上、ノードラベルを下に付ける。 2: ノードラベルを上、分割ラベルを下に付ける。 3: 枝に分割ラベルを付け、ノードにはラベルを付けない。 4: 3の結果にノードラベルを付ける。
extra=0	ノードに情報を表示する。 0: 何の情報も表示しない。 1: ノードに含まれる個体数を表示する。 2: 正しく分類された比率を分数で示す。 3: 誤分類された比を示す。 4: クラス内の確率を示す。 5: クラス内の確率のみを示す。 100以上はパーセンテージ情報を示す。
under=FALSE	TRUEのときは extra で指定した情報をボックスの下に示す。
fallen.leaves=FALSE	TRUEのとき葉の位置を揃える。 ただし枝が垂直になる。
uniform=TRUE	TRUEのとき枝の長さは同じである。 FALSEの場合は当てはめ値に比例する。
digits=2	小数点以下の桁数をコントロールする。

varlen=-8	分割する変数の文字列の長さをコントロールする。 デフォルト値では8文字以上は切り捨てる。
faclen=3	分割ラベルをコントロールする。 デフォルト値では3文字以上は切り捨てる。
cex=NULL tweak=1	文字列のサイズをコントロールする。少数で指定する。 デフォルトは cex=1 である。cex、tweakどちらでもよい。

引数のオプションを調整し、木を作成した4つの例を次に示し、その結果を図6に示す。

```
> install.packages("rpart.plot")
> library(rpart.plot)
> temp.par<-par(mfrow=c(2,2))
> rpart.plot(res, main="A")
> rpart.plot(res, extra=1, under=TRUE,main="B")
> rpart.plot(res, extra=1, type=1, under=TRUE, nn=TRUE,main="C")
> rpart.plot(res, extra=2, type=2, shadow.col="gray",main="D")
> par(temp.par)
```

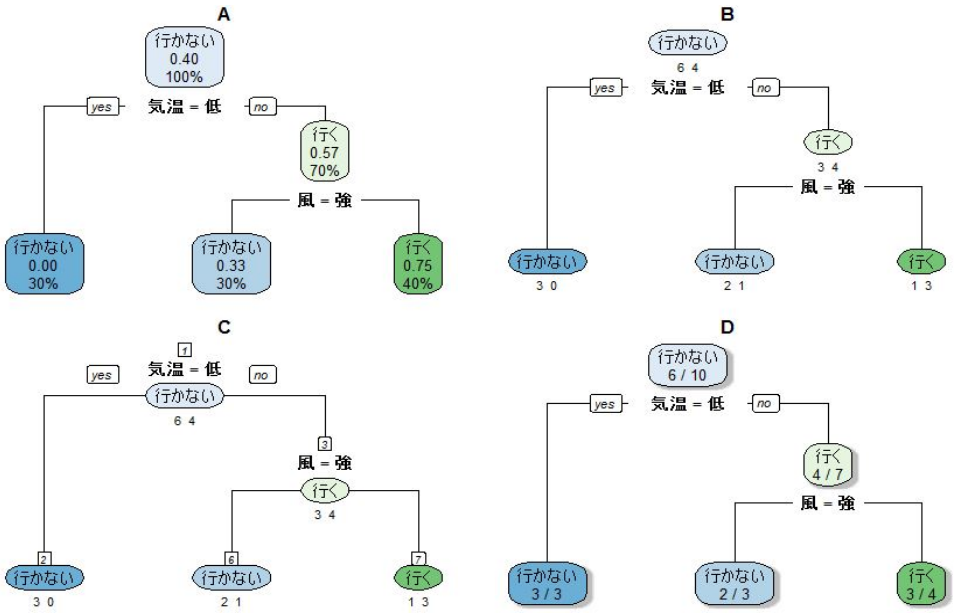
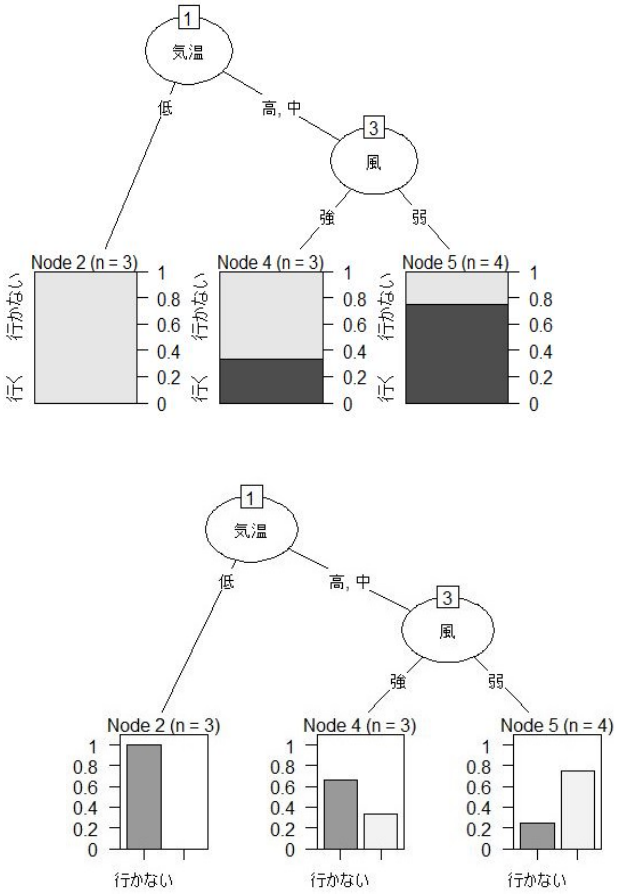


図6　関数 part.plot による木の図示例

(3)　関数 plot.party による図示

パッケージ partykit の中の変数 as.party で変換したオブジェクトを用いると、関数 plot で図7(a)のような木を作成することができる。すでに変換した res3 を用いた例を次に示し、作成したグラフを図7(a)に示す。ここで用いた plot は plot.party である。関数 plot.party では、葉の部分を装飾する引数 terminal_panel がある。つづいて、棒グラフを横並べする例を次に示し、その結果を図7(b)に示す。

```
> install.packages("partykit")
> library(partykit)
> plot(res3)                                    #図7(a)の作成
> bc<- c("gray60","gray95")       #2つの棒グラフの色を指定
> plot(res3, terminal_panel =node_barplot(res3,fill=bc, beside =TRUE))       #図(b)の作成
```



(a) (b)

図7　関数 plot.party による分類木

(4)　関数 fancyRpartPlot による図示

関数 fancyRpartPlot はパッケージ rattle の中の関数である。関数 rpart で作成した結果を直接用いることができる。次に rpart の結果を用いて、グラフを作成する例を示す。関数 fancyRpartPlot で作成した木は、自動的にルートおよび葉を含むノードを異なる色で塗りつぶしているので見やすい。

```
> install.packages(c("rattle", "RColorBrewer"))
> library("rattle")
> fancyRpartPlot(res)
```

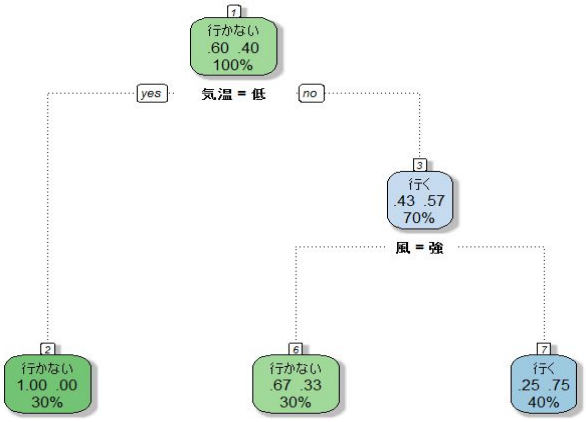


図8　関数 fancyRpartPlot による分類木

3.3　木の生長のコントロールと剪定

ツリーモデルは、木の枝を茂らせることにより、用いたデータ (学習データ) の予測・判別の精度を向上させることができる。ただし、そのモデル作成に用いていないデータ (テストデータ) について精度がよいという保証はない。よって、ツリーモデルでデータを当てはめるときには、木の生長をコントロール、あるいは何らかの基準で成長しすぎた木を剪定することが必要である。これは果樹園の剪定作業の考え方と同じである。一本の木の果実の数が多いほど経済効果がよいとは限らない。数が多いと果実の品質が落ちるため、果実の質をコントロールするために剪定作業を行う。

(1) 木の生長のコントロール

関数 rpart では、生成される木が成長しすぎないように事前に生長をコントロールする関数 rpart.control に基づいてモデルを構築する。関数 rpart.control のデフォルトの指定を次に示す。

```
rpart.control(minsplit = 20, minbucket = round(minsplit/3), cp = 0.01, maxcompete = 4,
maxsurrogate = 5, usesurrogate = 2, xval = 10, surrogatestyle = 0, maxdepth = 30, ...)
```

関数 rpart.control の中の引数は、関数 rpart の中で独立した引数として用いることが可能である。主なオプションを表6に示す。表に示していない、maxcompete、maxsurrogate、usesurrogate、surrogatestyle は分割するときに競合や代理分割の情報を保持する数である。この結果は分割について詳細に分析する際に用いるが、通常はデフォルトのままでもよい。これらの結果は、summary から確認できる。詳細は help(rpart.control) を参照してください。

表6　引数 control のオプション

引数とデフォルト	オプションの内容
minsplit = 20	ノードにおける最小数の個体数
minbucket= round(minsplit/3)	葉における最小数の個体数
cp = 0.01	複雑度。木の生長をコントロールする。
xval = 10	交差確認の数
maxdepth = 30	木の深さの最大数。

木の生長をコントロールする重要な引数は minsplit、minbucket 、cp である。引数 minsplit では分割する際にノード (葉以外) に含まれる個体数の最小値を指定する。例1の海水浴のデータの個体数は10である。よってデフォルトまま rpart (海水浴~.,dat) を実行すると木が生成されない。これは、データの個体数がデフォルトの値 minsplit = 20 より小さいからである。引数 minsplit の値を20, 8, 5にした結果を次に示す。

```
> rpart(海水浴~.,dat)
n= 10

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

1) root　10　4　行かない (0.6000000　0.4000000) *

> rpart(海水浴~.,dat,minsplit=8)
n= 10 node), split, n, loss, yval, (yprob)
* denotes terminal node

1) root 10 4 行かない (0.6000000 0.4000000)
2) 気温=低 3 0 行かない (1.0000000 0.0000000) *
3) 気温=高,中 7 3 行く (0.4285714 0.5714286) *

> rpart(海水浴~.,dat,minsplit=5)
n= 10

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

```
1) root 10 4 行かない (0.6000000 0.4000000)
  2) 気温=低 3 0 行かない (1.0000000 0.0000000) *
  3) 気温=高,中 7 3 行く (0.4285714 0.5714286)
    6) 風=強 3 1 行かない (0.6666667 0.3333333) *
    7) 風=弱 4 1 行く (0.2500000 0.7500000) *
```

引数 minbucket では、葉に含む個体数の最小値を指定する。デフォルトでは minsplit の約三分の一の整数を指定している。たとえば、例1では minsplit = 3 を用いているので、葉に含む標本の最小値は1になる

引数 cp では複雑度という統計量を指定する。複雑度 cp の値が小さいほど木の枝が多くなる。関数 rpart はこれらの引数の中、任意の一つの条件を満たさないと木の生長を止める。

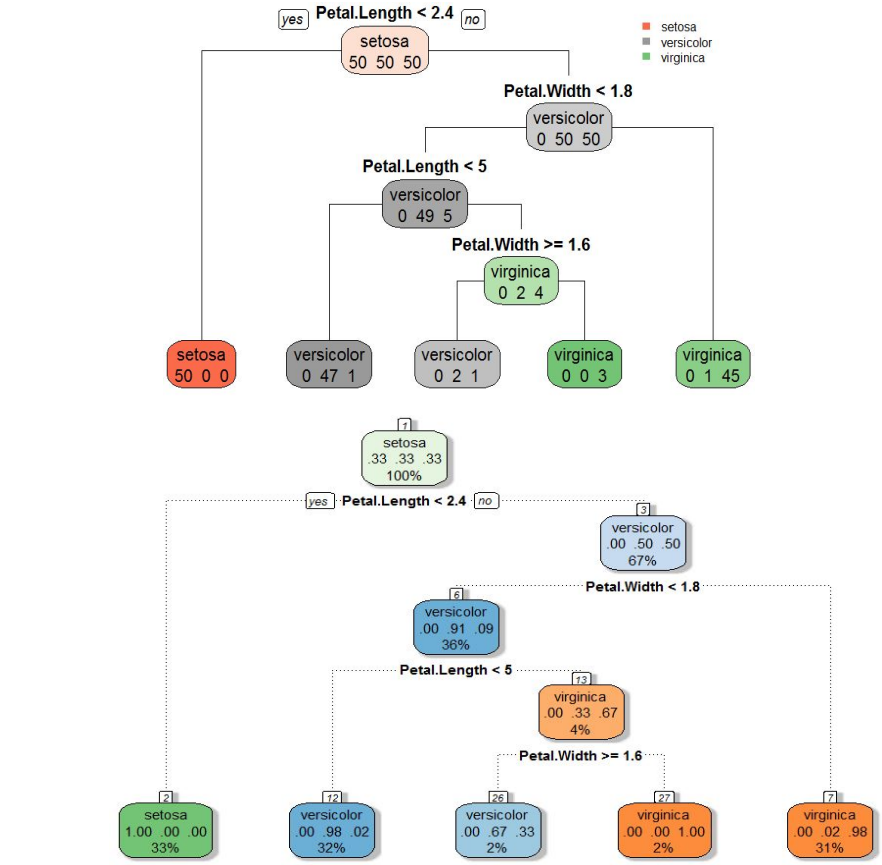
(2) 木の剪定

関数 rpart は木を成長させると同時に、交差確認法の結果も計算する。関数 printcp でその結果を返すことができる。また関数 plotcp は、複雑パラメター cp (complexity parameter) と木の深さの関係をグラフで示す。本章では複雑度グラフとよぶことにする。

例3 データ iris の最大木を生成し、選定に必要となる cp を求め、Min-1SE ルールで選定せよ。

まず、以下のように木の生長をコントロールする引数の条件を緩め、最大の木を生成する。

```
> set.seed(10)
> iris.rp<-rpart(Species~.,iris,minsplit=5,cp=0)     #最大木を作成
> rpart.plot(iris.rp,type=1,extra=1,cex=1.4)          #図9(a)の作成
> fancyRpartPlot(iris.rp)                                    #図9(b)の作成
```



(a) (b)

図9 データ iris の分類木

関数 rintcp で選定に関連する複雑度などを確認することができる。

```
> printcp(iris.rp)

Classification tree:
rpart(formula = Species ~ ., data = iris, minsplit = 5, cp = 0)

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

Root node error: 100/150 = 0.66667
n= 150

    CP nsplit rel error xerror    xstd
1 0.50      0      1.00   1.16 0.051277
2 0.44      1      0.50   0.72 0.061188
3 0.02      2      0.06   0.07 0.025833
4 0.01      3      0.04   0.08 0.027520
5 0.00      4      0.03   0.07 0.025833
```

CP 列が複雑度、nsplit は木のサイズ、rel error は誤分類率、xerror は交差確認法による誤分類率の平均値、xstd は交差確認法の誤分類率の標準偏差である。木の剪定方法としては主に2つの基準が用いられている。

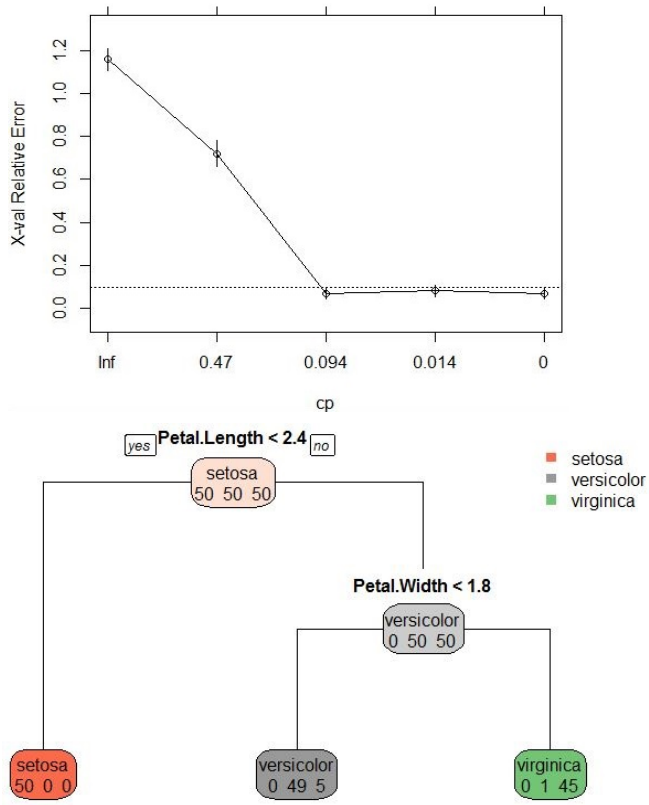
基準1. 交差確認法による誤分類率 xerror が最も小さい値。
基準2. 交差確認における xerror の最小値に1倍の標準偏差を足した値 (Min - 1SEルール)。

標本サイズが大きいときには基準1に基づいた方法をすすめる。標本サイズが小さいときには、交差確認の結果のバラツキが大きいので、後者の方法を用いた方がよい。後者は比較的保守的な考えである。

交差確認の誤分類率による基準1で判断すると nsplit は4になり、基準2である Min-1SE ルールを用いると 0.07 + 0.025833 = 0.095833 になる。これをもっともはやく下回っている xerror が 0.07、nsplit = 2、CP = 0.02 の行である。よって、Min-1SE ルールに基づいた木のコントロールや剪定は CP = 0.02 を用いればよい。

出力された cp, nsplit, xerror の値をグラフで示す関数 plotcp がある。その例を次に示す。下側の横軸のメモリは cp 値、上側の横軸は木のサイズ (分割の回数+1)、縦軸は xerror の値である。xerror は交差確認法による誤分類の比率である。よって、計算を繰り返しても、同じの結果が得られるとは限らない。点線の横線は xerror の誤分類の最小値に1倍の標準偏差を足した値である。

```
plotcp(iris.rp)
```



(a) モデル iris.rp の複雑度グラフ　　(b) 剪定した分類木

図10　複雑度グラフと剪定した分類木

Min-1SE を基準とした場合、点線をいち早く下回ったところまで用いる。よって、0.014 < cp < 0.94 で剪定すればよい。木の剪定は cp 値を用いて木を作成しなおす方法と剪定を行う関数 prune を用いる方法がある。どの方法を用いても結果同じである。次に剪定の例を示し、その結果を 8(b) に示す。

```
> iris.rp1<-prune(iris.rp, minsplit=5,cp=0.02)    #作成したモデルを剪定する
> rpart.plot(iris.rp1,type=1,extra=1)    #剪定した分類木の作成
```

複雑度 cp の計算式は $cp = R(t) - R(T)$ である。式の中の $R(t)$ はノード t における誤判別率、 $R(T)$ はノード t で分岐される左右の葉の誤判別率の和である。

図 10(b) を用いて説明すると、分岐ノードが一つである場合は、ルートの誤判別率は $R(t) = 1$, 左の枝の誤判別率は 0、右の誤判別率は 50/100 であり、$R(T) = 0 + 0.5$ である。よって $cp = 1 - 0.5 = 0.5$ になる。その下のノードには 100 個の個体がすべて versicol に判定されているので誤判別率 $R(t) = 50/100$ である。そのノード下の2つ葉における誤判別率の和は $R(T) = 5/100 + 1/100 = 6/100$ であり、cp は 50/100 - 6/100 = 0.44 である。図 9(a) を見ながら続けて計算すると次のノードの cp は5/100 - 1/100 - 2/100 = 0.02 になり、その下の cp は 2/100 - 1/100 - 0/100 = 0.0 になる。

3.4　ツリーモデルによる予測

作成したツリーモデルを用いて未知のデータについて予測・判別を行うためには、モデルの作成に用いていないデータが必要である。予測を行う関数は predict.rpart であり、略して predict で用いることが可能である。書式は回帰分析や一般化線形モデルの予測値を求める predict とほぼ同じである。

例4　iris データの五分の一を抽出し、予測用 (テスト用) のデータとした、予測結果を求めよ。

まず、学習用データとテスト用データを作成し、学習結果を考察する。

```
> set.seed(0)
> samp<-sample(1:150,30)       #30個の標本番号をランダムサンプリング
> train<-iris[-samp,]                #学習用のデータを作成する
> test<-iris[samp,]                  #テスト用のデータを作成する
> tr.rp<-rpart(Species~.,train, minsplit = 5, cp = 0)       #最大の木を生成する
> plotcp(tr.rp)                          #グラフを作成し，剪定の必要性を考察する
```

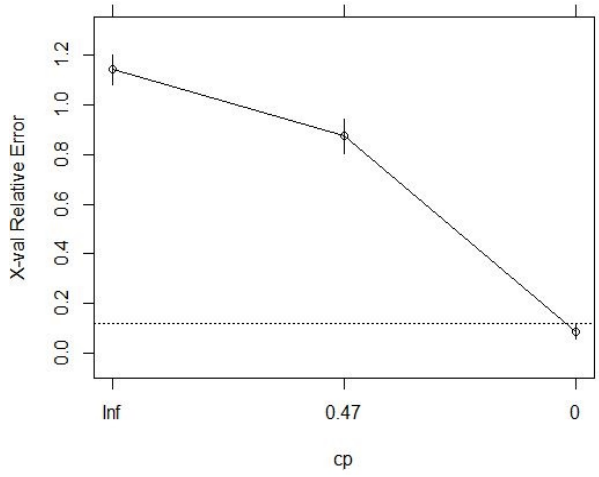


図11　モデル tr.rp の複雑度グラフ

上記の結果から、さらに剪定する必要がないと判断し、作成したモデルを用いて予測・判別を行う。予測の書式の例を次に示す。tr.rp は作成したモデル、test は予測に用いるデータである。返されたデータは各クラスに属する確率である。1行の135番は setosa に分類される確率は0、 versicolor に分類される確率は約 0.93、virginica に分類される確率は0.0698である。分類の問題では、確率がもっとく高いクラスに分類する。よって、iris データの135番は versicolor に分類される。

```
> predict(tr.rp, test)

      setosa     versicolor      virginica
135        0     0.93023256     0.06976744
40         1     0.00000000     0.00000000
56         0     0.93023256     0.06976744
<後略>
```

引数 type = "class" を付け加えると分類されるクラスのラベルを返す。

```
> te.rp<-predict(tr.rp, test, type="class")
> head(te.rp)

         135       40           56           85          133       30
  versicolor   setosa   versicolor   versicolor   virginica   setosa
```

上の番号はサンプリングされた個体の番号であり、その下のラベルは分類されたクラスのラベルである。確率データで説明したとおり、iris データの135番は versicolor に属する。このような分類結果と元のデータとの食い違いを考察するためには、次のように混同表を作成して考察した方がよい。

```
> table(test[,5],te.rp)              #予測結果の混同行列を作成

                te.rp
              setosa   versicolor   virginica
setosa            12            0           0
versicolor         0            9           0
```

```
virginica          0          2          7
```

返された結果から分かるように、誤分類率は 2/30 = 0.0667 であり、正解率は 1 - 0.0667 = 0.933 である。

ツリーモデルでは、モデルの作成に寄与している変数の重要度を求めることができる。パッケージ rpart には、重要度は $variable.importance に記録されている。分類にもっとも重要な変数は Petal.Width であり、その次が Petal.Length であることがわかる。

```
> tr.rp$variable.importance

 Petal.Width Petal.Length Sepal.Length Sepal.Width
    72.41989     67.27067     43.62303    32.18124
```

関数オブジェクト rpart のクラスに属する主な関数を表7に示す。

表7　rpart クラスの主な関数

関数名	主な機能
rpart	木を生成する
plot.rpart	木を図示する。略してplot。
text.rpart	木にラベルを追加。略してtext。
plotcp	複雑度を図示する。
predict	モデルで予測する。
print.rpart	ルールの出力。略してprint。
printcp	複雑度の詳細を出力する。
prune.part	木を剪定する。略してprune。
residuals.rpart	残差を返す。略してresiduals。
rpart.control	コントロールオプションを変更する。
summary.rpart	要約を返す。summary
$variable.importance	変数の重要度を返す。

4　回帰木

CART において回帰木の分割基準となる不純度は、実測値 y_{ik} と区間 k の分散である。式の中の $\bar{y}_k$ は区間 k の平均、$|K|$ は区間 k の中の個体数である。

$$D = \frac{1}{|K|}\sum_{k}(y_{ik} - \bar{y}_k)^2$$

関数 rpart による回帰モデルの作成について実データを用いた例を説明する。

パッケージ rpart の中にデータセット cu.summary がある。データセット cu.summary は1990年アメリカで117台の車について値段 (Price)、生産国 (Country)、信頼性 (Reliability)、燃費 (Mileage)、車種 (Type) について記録したデータフレームである。データ cu.summary の要約を次に示す。

```
> summary(cu.summary)

     Price          Country      Reliability       Mileage         Type
 Min.   : 5866   USA      :49   Much worse: 18   Min.   :18.00   Compact: 22
 1st Qu.:10125   Japan    :31   worse     : 12   1st Qu.:21.00   Large  :  7
 Median :13150   Germany  :11   average   : 26   Median :23.00   Medium : 30
 Mean   :15743   Japan/USA:  9   better    :  8   Mean   :24.58   Small  : 22
 3rd Qu.:18900   Korea    :  5   Much better: 21   3rd Qu.:27.00   Sporty : 26
 Max.   :41990   Sweden   :  5   NA's      : 32   Max.   :37.00   Van    : 10
                 (Other):  7                     NA's   :57
```

例5　車の値段 Price を被説明変数とし、関数 rpart を用いて回帰木を作成し考察せよ。
　変数 Price を目的変数、その残りを説明変数とした回帰木を次のように作成する。

```
> set.seed(0)
> cars.rp<-rpart(Price~Country+Reliability+Mileage+Type,data= cu.summary)
```

　作成した回帰木について剪定が必要であるかを見るため、関数 plotcp で考察する。返された図を見ると Min + 1SE の線の下方に合計4つのノードがある。よって、選定が必要である。もし、Min - 1SE ルールで剪定すると木のサイズは4になり、それに対応する cp は 0.074 である。

```
> plotcp(cars.rp)
```

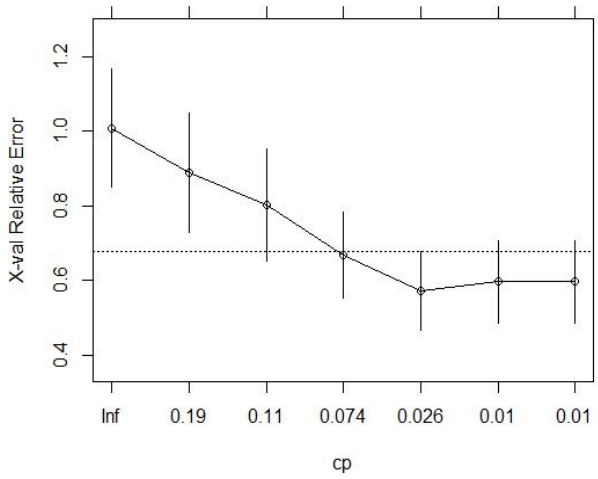


図12　モデル cars.rp の複雑度グラフ

　複雑度 cp = 0.07 で剪定し、パッケージ partykit の関数を利用してルールと回帰木のグラフを作成する例を次に示す。

```
> cars.rp2<-prune(cars.rp,cp=0.07)
> cars.pa<-as.party(cars.rp2)
> print(cars.pa)

Model formula:
Price ~ Country + Reliability + Mileage + Type

Fitted party:
[1] root
|   [2] Type in Compact, Small, Sporty, Van
|   |   [3] Country in Brazil, France, Japan, Japan/USA, Korea, Mexico, USA: 11555.159 (n = 69,
err = 1426420615.2)
|   |   [4] Country in Germany, Sweden: 22317.727 (n = 11, err = 797004218.2)
|   [5] Type in Large, Medium
|   |   [6] Country in France, Korea, USA: 18697.280 (n = 25, err = 1021101575.0)
|   |   [7] Country in England, Germany, Japan, Sweden: 27646.000 (n = 12, err = 558955038.0)

Number of inner nodes: 3
Number of terminal nodes: 4

> plot(cars.pa)
```

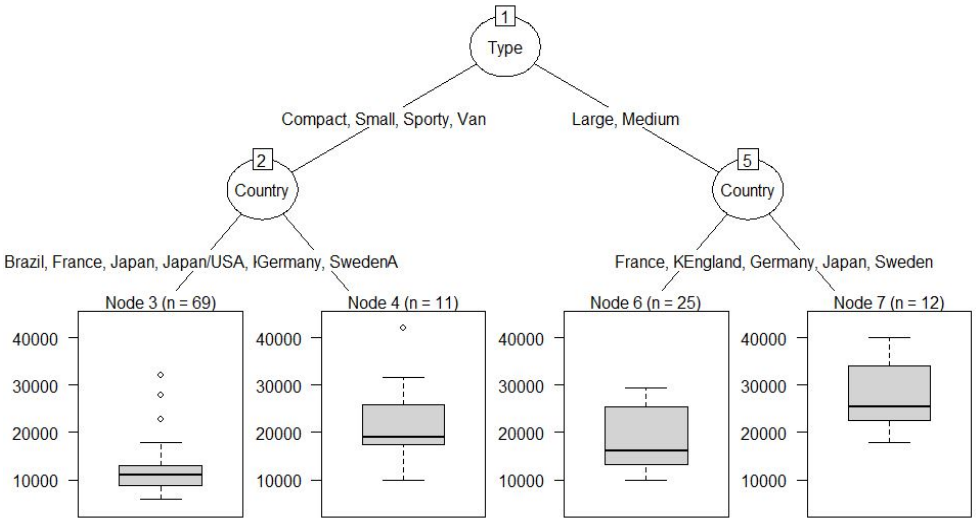


図13　車値段に関す回帰木

この回帰木は車の値段を4つのルール当てはめているので、非常に粗いことは言うまでもない。モデルの残差は residuals で返すことができる。モデルの残差の二乗和を次に示す。ついでに、線形回帰モデルによる残差二乗和も示す。このデータにおいては、ツリーモデルは重回帰モデルより当てはめがよくない。

```
> sum(residuals(cars.rp)^2)
[1] 3183887398
> cars.lm<-lm(Price~Type+Country,data= cu.summary)
> sum(residuals(cars.lm)^2)
[1] 3086927875
```

被説明変数がカウントデータの場合はポアソン分布に基づいた回帰木を用いて分析することができる。関数 rpart 引数 method = "poisson" を追加するだけである。

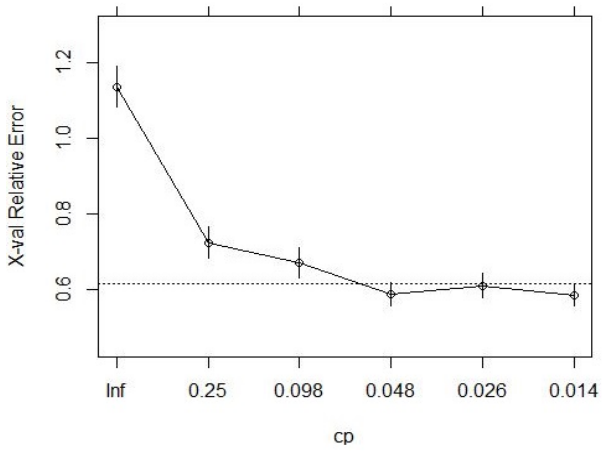
例6　ポアソン回帰分析の章で用いたデータ data (esoph) について回帰木を求め、一般化線形モデルの結果と比較せよ。

関数 glm と rpart による回帰モデルを次のように求める。

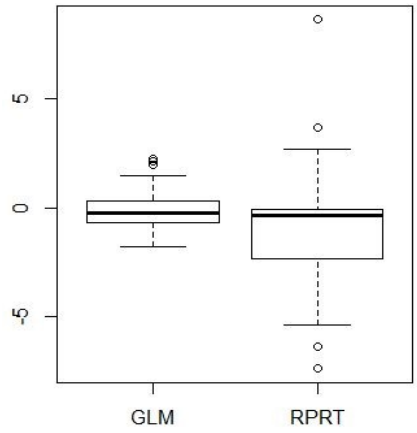
```
> data(esoph)
> eso.m<-glm(ncases~agegp+alcgp+tobgp,offset=log(ncontrols),data=esoph, family=poisson)
# 一般化線形モデルによるあてはめ
> eso.rp<-rpart(ncases~agegp+alcgp+tobgp,weights=ncontrols,data=esoph, method="poisson")
# ツリーモデルによるあてはめ
> plotcp(eso.rp)          # 複雑度グラフ図14(a)
```

複雑度グラフを見ると剪定しなくてよさそうである。次に両モデルの残差を比較するため、残差の箱ひげ図を次のように作成し、その結果を図 13(b) に示す。箱ひげ図から分かるように、残差の中央値には明らかな差がないが、一般化線形モデルによるあてはめの残差のバラツキが小さい。

```
> boxplot(data.frame(GLM=residuals(eso.m),RPRT=residuals(eso.rp)))
```



(a) モデル eso.rp の複雑度グラフ



(b) 両モデルの残差の箱ひげ図

図14 複雑度と残差の箱ひげ図

5 その他

Rに実装されている関数は CART 流の rpart 以外にパッケージ tree の中の関数 tree、パッケージ caret の中の ctree、パッケージ RWeka の中には C4.5 を発展させた J48、パッケージ C50 の中には関数 C5.0 などいくつかの関数がある。分類木としては C5.0 の分類精度よい。ただし、関数 C5.0 は回帰木の機能はない。つまり、目的変数がカテゴリカルデータに限っている。関数 C5.0 のデータ分割に用いている指標は情報利得比である。情報利得比は、エントロピーの比である。使用方法は rpart と類似している。次にその例を示す。

```
> install.packages("C50"); library(C50)
> m1<-C5.0(Species~.,iris)
> summary(m1)

Call:  C5.0.formula(formula = Species ~ ., data = iris)

C5.0 [Release 2.07 GPL Edition]        Sun  Mar  26  17:49:28  2017
-------------------------------

Class  specified  by  attribute  `outcome'
Read  150  cases  (5  attributes)  from  undefined.data

Decision  tree:

Petal.Length <= 1.9: setosa (50)
Petal.Length > 1.9:
:...Petal.Width > 1.7: virginica (46/1)
    Petal.Width <= 1.7:
    :...Petal.Length <= 4.9: versicolor (48/1)
        Petal.Length > 4.9: virginica (6/2)

Evaluation on training data (150 cases):

        Decision  Tree
      ----------------
    Size      Errors

       4    4( 2.7%)   <<

     (a)   (b)   (c)    <-classified as
```

```
----    ----    ----
 50                      (a): class setosa
         47      3       (b): class versicolor
          1     49       (c): class virginica

        Attribute usage:

        100.00% Petal.Length
        66.67% Petal.Width

Time:   0.0 secs
```

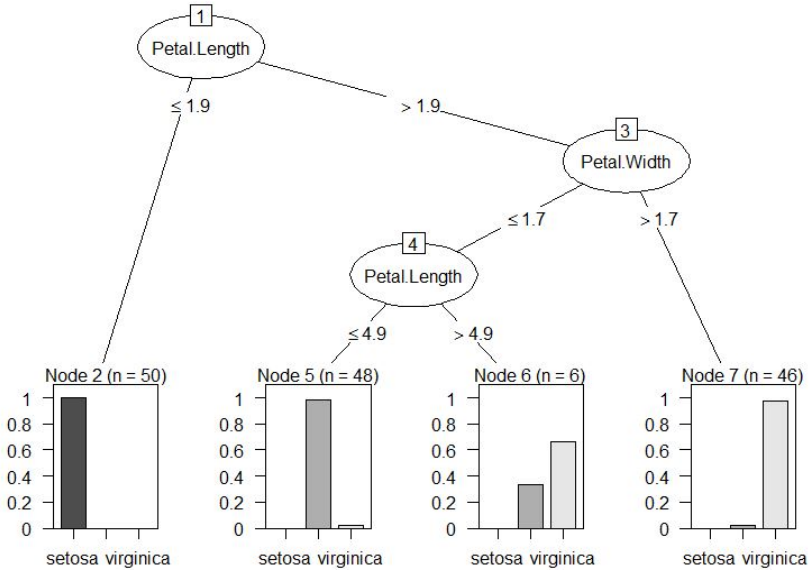


図15　関数 C5.0 の分類木

関数 C5.0 は、変数が多いときには強大な木を出力することになる。数十個の変数であっても、画面上に表示しきれない木になることがある。

パッケージ RWeka の中の J48 は、C4.5 と C5.0 の中間バージョンである。よって通常 C5.0 とほぼ同じである。回帰木や分類木は理解やすいが、特に回帰問題では精度がよいものとは言いがたい。パフォマンスを引き上げる方法を次の章で紹介するツリーモデルに関しては下川・他 (2013) が詳しい。

参考文献

下川敏雄・杉本知之・後藤昌司：樹木構造接近法,共立出版 (2013)