

# Package ‘C50’

June 7, 2014

**Type** Package

**Title** C5.0 Decision Trees and Rule-Based Models

**Version** 0.1.0-19

**Date** 2014-06-5

**Author** Max Kuhn, Steve Weston, Nathan Coulter. C code for C5.0 by R. Quinlan

**Maintainer** Max Kuhn <mxkuhn@gmail.com>

**Description** C5.0 decision trees and rule-based models for pattern recognition

**Depends** R (>= 2.10.0)

**License** GPL-3

**LazyLoad** yes

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2014-06-07 08:13:11

## R topics documented:

C5.0.default . . . . .	2
C5.0Control . . . . .	4
C5imp . . . . .	6
churn . . . . .	7
predict.C5.0 . . . . .	8
summary.C5.0 . . . . .	10

<b>Index</b>	<b>12</b>
--------------	-----------

C5.0.default

C5.0 Decision Trees and Rule-Based Models

**Description**

Fit classification tree models or rule-based models using Quinlan's C5.0 algorithm

**Usage**

```
C5.0(x, ...)

## Default S3 method:
C5.0(x, y, trials = 1, rules= FALSE,
     weights = NULL,
     control = C5.0Control(),
     costs = NULL, ...)

## S3 method for class 'formula'
C5.0(formula, data, weights, subset,
     na.action = na.pass, ...)
```

**Arguments**

x	a data frame or matrix of predictors.
y	a factor vector with 2 or more levels
trials	an integer specifying the number of boosting iterations. A value of one indicates that a single model is used.
rules	A logical: should the tree be decomposed into a rule-based model?
weights	an optional numeric vector of case weights. Note that the data used for the case weights will not be used as a splitting variable in the model (see <a href="http://www.rulequest.com/see5-win.html#CASEWEIGHT">http://www.rulequest.com/see5-win.html#CASEWEIGHT</a> for Quinlan's notes on case weights).
control	a list of control parameters; see <a href="#">C5.0Control</a>
costs	a matrix of costs associated with the possible errors. The matrix should have C columns and rows where C is the number of class levels.
formula	a formula, with a response and at least one predictor.
data	an optional data frame in which to interpret the variables named in the formula.
subset	optional expression saying that only a subset of the rows of the data should be used in the fit.
na.action	a function which indicates what should happen when the data contain NAs. The default is to include missing values since the model can accommodate them.
...	other options to pass into the function (not currently used with default method)

## Details

This model extends the C4.5 classification algorithms described in Quinlan (1992). The details of the extensions are largely undocumented. The model can take the form of a full decision tree or a collection of rules (or boosted versions of either).

When using the formula method, factors and other classes are preserved (i.e. dummy variables are not automatically created). This particular model handles non-numeric data of some types (such as character, factor and ordered data).

The cost matrix should be  $C \times C$ , where  $C$  is the number of classes. Diagonal elements are ignored. Columns should correspond to the true classes and rows are the predicted classes. For example, if  $C = 3$  with classes Red, Blue and Green (in that order), a value of 5 in the (2,3) element of the matrix would indicate that the cost of predicting a Green sample as Blue is five times the usual value (of one). Note that when costs are used, class probabilities cannot be generated using `predict.C5.0`.

Internally, the code will attempt to halt boosting if it appears to be ineffective. For this reason, the value of `trials` may be different from what the model actually produced. There is an option to turn this off in `C5.0Control`.

## Value

An object of class `C5.0` with elements:

<code>boostResults</code>	a parsed version of the boosting table(s) shown in the output
<code>call</code>	the function call
<code>caseWeights</code>	not currently supported.
<code>control</code>	an echo of the specifications from <code>C5.0Control</code>
<code>cost</code>	the text version of the cost matrix (or <code>""</code> )
<code>costMatrix</code>	an echo of the model argument
<code>dims</code>	original dimensions of the predictor matrix or data frame
<code>levels</code>	a character vector of factor levels for the outcome
<code>names</code>	a string version of the names file
<code>output</code>	a string version of the command line output
<code>predictors</code>	a character vector of predictor names
<code>rbm</code>	a logical for rules
<code>rules</code>	a character version of the rules file
<code>size</code>	n integer vector of the tree/rule size (or sizes in the case of boosting)
<code>tree</code>	a string version of the tree file
<code>trials</code>	a named vector with elements Requested (an echo of the function call) and Actual (how many the model used)

## Note

The command line version currently supports more data types than the R port. Currently, numeric, factor and ordered factors are allowed as predictors.

**Author(s)**

Original GPL C code by Ross Quinlan, R code and modifications to C by Max Kuhn, Steve Weston and Nathan Coulter

**References**

Quinlan R (1993). C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, <http://www.rulequest.com/see5-unix.html>

**See Also**

[C5.0Control](#), [summary.C5.0](#), [predict.C5.0](#), [C5imp](#)

**Examples**

```
data(churn)

treeModel <- C5.0(x = churnTrain[, -20], y = churnTrain$churn)
treeModel
summary(treeModel)

ruleModel <- C5.0(churn ~ ., data = churnTrain, rules = TRUE)
ruleModel
summary(ruleModel)
```

---

C5.0Control

*Control for C5.0 Models*


---

**Description**

Various parameters that control aspects of the C5.0 fit.

**Usage**

```
C5.0Control(subset = TRUE,
            bands = 0,
            winnow = FALSE,
            noGlobalPruning = FALSE,
            CF = 0.25,
            minCases = 2,
            fuzzyThreshold = FALSE,
            sample = 0,
            seed = sample.int(4096, size = 1) - 1L,
            earlyStopping = TRUE,
            label = "outcome")
```

**Arguments**

subset	A logical: should the model evaluate groups of discrete predictors for splits? Note: the C5.0 command line version defaults this parameter to FALSE, meaning no attempted groupings will be evaluated during the tree growing stage.
bands	An integer between 2 and 1000. If TRUE, the model orders the rules by their affect on the error rate and groups the rules into the specified number of bands. This modifies the output so that the effect on the error rate can be seen for the groups of rules within a band. If this options is selected and rules = FALSE, a warning is issued and rules is changed to TRUE.
winnow	A logical: should predictor winnowing (i.e feature selection) be used?
noGlobalPruning	A logical to toggle whether the final, global pruning step to simplify the tree.
CF	A number in (0, 1) for the confidence factor.
minCases	an integer for the smallest number of samples that must be put in at least two of the splits.
fuzzyThreshold	A logical toggle to evaluate possible advanced splits of the data. See Quinlan (1993) for details and examples.
sample	A value between (0, .999) that specifies the random proportion of the data should be used to train the model. By default, all the samples are used for model training. Samples not used for training are used to evaluate the accuracy of the model in the printed output.
seed	An integer for the random number seed within the C code.
earlyStopping	A logical to toggle whether the internal method for stopping boosting should be used.
label	A character label for the outcome used in the output.

**Value**

A list of options.

**Author(s)**

Original GPL C code by Ross Quinlan, R code and modifications to C by Max Kuhn, Steve Weston and Nathan Coulter

**References**

Quinlan R (1993). C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, <http://www.rulequest.com/see5-unix.html>

**See Also**

[C5.0](#), [predict.C5.0](#), [summary.C5.0](#), [C5imp](#)

## Examples

```
data(churn)

treeModel <- C5.0(x = churnTrain[, -20], y = churnTrain$churn,
  control = C5.0Control(winnow = TRUE))
summary(treeModel)
```

C5imp

*Variable Importance Measures for C5.0 Models*

## Description

This function calculates the variable importance (aka attribute usage) for C5.0 models.

## Usage

```
C5imp(object, metric = "usage", pct = TRUE, ...)
```

## Arguments

object	an object of class C5.0
metric	either 'usage' or 'splits' (see Details below)
pct	a logical: should the importance values be converted to be between 0 and 100?
...	other options (not currently used)

## Details

By default, C5.0 measures predictor importance by determining the percentage of training set samples that fall into all the terminal nodes after the split (this is used when `metric = "usage"`). For example, the predictor in the first split automatically has an importance measurement of 100 percent. Other predictors may be used frequently in splits, but if the terminal nodes cover only a handful of training set samples, the importance scores may be close to zero. The same strategy is applied to rule-based models as well as the corresponding boosted versions of the model.

There is a difference in the attribute usage numbers between this output and the nominal command line output. Although the calculations are almost exactly the same (we do not add 1/2 to everything), the C code does not display that an attribute was used if the percentage of training samples covered by the corresponding splits is very low. Here, the threshold was lowered and the fractional usage is shown.

When `metric = "splits"`, the percentage of splits associated with each predictor is calculated.

## Value

a data frame with a column Overall with the predictor usage values. The row names indicate the predictor.

**Author(s)**

Original GPL C code by Ross Quinlan, R code and modifications to C by Max Kuhn, Steve Weston and Nathan Coulter

**References**

Quinlan R (1993). C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, <http://www.rulequest.com/see5-unix.html>

**See Also**

[C5.0](#), [C5.0Control](#), [summary.C5.0](#), [predict.C5.0](#)

**Examples**

```
data(churn)

treeModel <- C5.0(x = churnTrain[, -20], y = churnTrain$churn)
C5imp(treeModel)
C5imp(treeModel, metric = "splits")
```

---

churn

---

*Customer Churn Data*


---

**Description**

A data set from the MLC++ machine learning software for modeling customer churn. There are 19 predictors, mostly numeric: state (categorical), account\_length, area\_code, international\_plan (yes/no), voice\_mail\_plan (yes/no), number\_vmail\_messages, total\_day\_minutes, total\_day\_calls, total\_day\_charge, total\_eve\_minutes, total\_eve\_calls, total\_eve\_charge, total\_night\_minutes, total\_night\_calls, total\_night\_charge, total\_intl\_minutes, total\_intl\_calls, total\_intl\_charge and number\_customer\_service\_calls.

The outcome is contained in a column called churn (also yes/no).

The training data has 3333 samples and the test set contains 1667.

A note in one of the source files states that the data are "artificial based on claims similar to real world".

A rule-based model shown on the RuleQuest website contains 19 rules, including:

```
Rule 1: (2221/60, lift 1.1)
  international plan = no
  total day minutes <= 223.2
  number customer service calls <= 3
  -> class 0 [0.973]
```

```
Rule 5: (1972/87, lift 1.1)
  total day minutes <= 264.4
```

```
total intl minutes <= 13.1
total intl calls > 2
number customer service calls <= 3
-> class 0 [0.955]
```

```
Rule 10: (60, lift 6.8)
international plan = yes
total intl calls <= 2
-> class 1 [0.984]
```

```
Rule 12: (32, lift 6.7)
total day minutes <= 120.5
number customer service calls > 3
-> class 1 [0.971]
```

This implementation of C5.0 contains the same rules, but the rule numbers are different than above.

### Usage

```
data(churn)
```

### Value

churnTrain	The training set
churnTest	The test set.

### Source

<http://www.sgi.com/tech/mlc/>, <http://www.rulequest.com/see5-examples.html>

---

predict.C5.0	<i>Predict new samples using a C5.0 model</i>
--------------	---

---

### Description

This function produces predicted classes or confidence values from a C5.0 model.

### Usage

```
## S3 method for class 'C5.0'
predict(object, newdata = NULL,
        trials = object$trials["Actual"],
        type = "class", na.action = na.pass, ...)
```



**Arguments**

object	an object of class <code>C5.0</code>
newdata	a matrix or data frame of predictors
trials	an integer for how many boosting iterations are used for prediction. See the note below.
type	either "class" for the predicted class or "prob" for model confidence values.
na.action	when using a formula for the original model fit, how should missing values be handled?
...	other options (not currently used)

**Details**

Note that the number of trials in the object may be less than what was specified originally (unless `earlyStopping = FALSE` was used in `C5.0Control`). If the number requested is larger than the actual number available, the maximum actual is used and a warning is issued.

Model confidence values reflect the distribution of the classes in terminal nodes or within rules.

For rule-based models (i.e. not boosted), the predicted confidence value is the confidence value from the most specific, active rule. Note that C4.5 sorts the rules, and uses the first active rule for prediction. However, the default in the original sources did not normalize the confidence values. For example, for two classes it was possible to get confidence values of (0.3815, 0.8850) or (0.0000, 0.922), which do not add to one. For rules, this code divides the values by their sum. The previous values would be converted to (0.3012, 0.6988) and (0, 1). There are also cases where no rule is activated. Here, equal values are assigned to each class.

For boosting, the per-class confidence values are aggregated over all of the trees created during the boosting process and these aggregate values are normalized so that the overall per-class confidence values sum to one.

When the `cost` argument is used in the main function, class probabilities derived from the class distribution in the terminal nodes may not be consistent with the final predicted class. For this reason, requesting class probabilities from a model using unequal costs will throw an error.

**Value**

when `type = "class"`, a factor vector is returned. When `type = "prob"`, a matrix of confidence values is returned (one column per class).

**Author(s)**

Original GPL C code by Ross Quinlan, R code and modifications to C by Max Kuhn, Steve Weston and Nathan Coulter

**References**

Quinlan R (1993). C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, <http://www.rulequest.com/see5-unix.html>

See Also

C5.0, C5.0Control, summary.C5.0, C5imp

Examples

```
data(churn)

treeModel <- C5.0(x = churnTrain[, -20], y = churnTrain$churn)
predict(treeModel, head(churnTest[, -20]))
predict(treeModel, head(churnTest[, -20]), type = "prob")
```

---

summary.C5.0	Summaries of C5.0 Models
--------------	--------------------------

---

Description

This function prints out detailed summaries for C5.0 models.

Usage

```
## S3 method for class 'C5.0'
summary(object, ...)
```

Arguments

object	an object of class C5.0
...	other options (not currently used)

Details

The output of this function mirrors the output of the C5.0 command line version.

The terminal nodes have text indicating the number of samples covered by the node and the number that were incorrectly classified. Note that, due to how the model handles missing values, the sample numbers may be fractional.

There is a difference in the attribute usage numbers between this output and the nominal command line output. Although the calculations are almost exactly the same (we do not add 1/2 to everything), the C code does not display that an attribute was used if the percentage of training samples covered by the corresponding splits is very low. Here, the threshold was lowered and the fractional usage is shown.

Value

A list with values

output	a single text string with the model output
comp2	the call to this function

**Author(s)**

Original GPL C code by Ross Quinlan, R code and modifications to C by Max Kuhn, Steve Weston and Nathan Coulter

**References**

Quinlan R (1993). C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, <http://www.rulequest.com/see5-unix.html>

**See Also**

[C5.0](#), [C5.0Control](#), [summary.C5.0](#), [C5imp](#)

**Examples**

```
data(churn)

treeModel <- C5.0(x = churnTrain[, -20], y = churnTrain$churn)
summary(treeModel)
```

# Index

## \*Topic **datasets**

churn, [7](#)

## \*Topic **models**

C5.0.default, [2](#)

C5.0Control, [4](#)

C5imp, [6](#)

predict.C5.0, [8](#)

summary.C5.0, [10](#)

C5.0, [5](#), [7](#), [10](#), [11](#)

C5.0 (C5.0.default), [2](#)

C5.0.default, [2](#)

C5.0Control, [2–4](#), [4](#), [7](#), [9–11](#)

C5imp, [4](#), [5](#), [6](#), [10](#), [11](#)

churn, [7](#)

churnTest (churn), [7](#)

churnTrain (churn), [7](#)

predict.C5.0, [3–5](#), [7](#), [8](#)

summary.C5.0, [4](#), [5](#), [7](#), [10](#), [10](#), [11](#)