Package: C50 (via r-universe)

November 23, 2026

Type Package

Title C5.0 Decision Trees and Rule-Based Models

Version 1.0.1

Maintainer Max Kuhn <mxkuhn@gmail.com>

Description C5.0 decision trees and rule-based models for pattern recognition that extend the work of Quinlan (1993, ISBN:1-55860-238-0).

License GPL-3

URL https://topepo.github.io/C5.0/, https://github.com/topepo/C5.0/

BugReports https://github.com/topepo/C5.0/issues

Depends R (>= 2.10.0)

Imports cli, Cubist (>= 0.3.0), partykit, rlang

Suggests covr, knitr, modeldata, rmarkdown, testthat (>= 3.0.0), tibble

VignetteBuilder knitr

Biarch true

Config/testthat/edition 3

Encoding UTF-8

LazyLoad yes

Config/roxygen2/version 8.1.1

NeedsCompilation yes

Author Max Kuhn [aut, cre], Steve Weston [ctb], Mark Culp [ctb], Nathan Coulter [ctb], Ross Quinlan [aut] (Author of imported C code), RuleQuest Research [cph] (Copyright holder of imported C code), Rulequest Research Pty Ltd. [cph] (Copyright holder of imported C code)

Repository https://cran.r-universe.dev

Date/Publication 2026-11-23 08:00:02 UTC

RemoteUrl https://github.com/cran/C50

RemoteRef HEAD

RemoteSha 276fbdc41c32a29e9ab6502f2a3284744fe15f0f

Contents

C5.0.default 2
C5.0Control 6
C5imp 7
plot.C5.0 9
predict.C5.0 10
summary.C5.0 12

Index **14**

`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

```
## 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. There is no upper limit, but fitting time, memory use, and model size grow linearly with the number of trials. When `earlyStopping = TRUE` in `C5.0Control()` (the default), boosting can stop before the requested number of trials; see the `trials` element of the returned object for the number actually 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 http://www.rulequest.com/see5-win.html#CASEWEIGHT for Quinlan's notes on case weights).
control	a list of control parameters; see C5.0Control()
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. Rows are the predicted classes and columns are the true classes. See the Costs section for which values are used and how.
...	other options to pass into the function (not currently used with default method)
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 NA. The default is to include missing values since the model can accommodate them.

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).

Predictors can be numeric, factor, ordered factor, or character columns. Character columns are treated as factors with sorted levels. Names and values are used exactly as given: there are no reserved names or special values, so a column can be called `"case weight"`, and a factor level can be `"?"`, `"N/A"`, or an empty string. The only exception is that no predictor can have the same name as the outcome label (`label` in `C5.0Control()`). Missing values (`NA`) are allowed in the predictors; C5.0 handles them directly. Rows with a missing outcome are not used.

Value

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

boostResults	a parsed version of the boosting table(s) shown in the output
call	the function call
caseWeights	a logical: were case weights used?
control	an echo of the specifications from C5.0Control()
cost	the text version of the cost matrix (or "" when no off-diagonal cost differs from 1)
costMatrix	the cost matrix, with its rows and columns in the order of the class levels

dims	original dimensions of the predictor matrix or data frame
levels	a character vector of factor levels for the outcome
names	a string version of the names file. The C code does not use it; it is kept for code that reads the predictor types and levels from it.
output	a string version of the command line output
predictors	a character vector of predictor names
rbm	a logical for rules
rules	a character version of the rules file
size	n integer vector of the tree/rule size (or sizes in the case of boosting)
spec	a description of the predictors and the outcome (names, types, and levels) that predict.C5.0() uses to prepare new data
format_version	the version of the data interface; 2 for models fitted with this version of the package
tree	a string version of the tree file
trials	a named vector with elements Requested (an echo of the function call) and Actual (how many the model used)

Costs

The cost matrix should be CxC, where C is the number of classes. 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).

The row and column names must both be the levels of y. They can be in any order: rows are matched by the row names and columns by the column names. If the matrix has no dimnames, the levels of y are used, in order, with a warning.

Which values are used:

- Diagonal elements are ignored.
- An off-diagonal value of 1 is the default cost, so it has no effect.
- All other off-diagonal values are used, including values between 0 and 1.
- A value of 0 (an error with no cost) is allowed when there are three or more classes. With two classes, both off-diagonal values must be greater than 0.
- Negative, missing, and infinite values are errors.

How the costs are used depends on the number of classes:

- With two classes, the training cases are reweighted so that cases of the class whose errors cost more count for more. These weights are multiplied by any case weights, and minCases is lowered to match.
- With any number of classes, the predicted class is the one with the lowest expected cost, not the most probable one. Because of this, class probabilities cannot be generated with predict.C5.0() when costs are used.

- For rule-based models with three or more classes, the costs are also used when the rules are formed.

Up to version 0.2.0, off-diagonal costs that were not greater than 1 were dropped without a message, so that C5.0 used a cost of 1 instead (a value of 0.5, for example, had no effect). The row names of the matrix were also ignored, and the rows were assumed to be in the order of the column names. Models fitted with such cost matrices will differ from those versions.

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()`.

Note

The command line version currently supports more data types than the R port. Currently, numeric, factor, ordered factor, and character columns 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

```
library(modeldata)
data(mlc_churn)

treeModel <- C5.0(x = mlc_churn[1:3333, -20], y = mlc_churn$churn[1:3333])
treeModel
summary(treeModel)

ruleModel <- C5.0(churn ~ ., data = mlc_churn[1:3333, ], 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",
  strip_time_stamps = TRUE
)
```

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.
strip_time_stamps	A logical to toggle whether timestamps should be removed from the C5.0 output. This removes the date/time from the header line and the "Time:" line showing execution time. Default is TRUE to make output reproducible.

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

```
library(modeldata)
data(mlc_churn)

treeModel <- C5.0(x = mlc_churn[1:3333, -20],
                  y = mlc_churn$churn[1:3333],
                  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

```
library(modeldata)
data(mlc_churn)

treeModel <- C5.0(x = mlc_churn[1:3333, -20], y = mlc_churn$churn[1:3333])
C5imp(treeModel)
C5imp(treeModel, metric = "splits")
```

plot.C5.0 *Plot a decision tree*

Description

Plot a decision tree.

Usage

```
## S3 method for class 'C5.0'
plot(x, trial = 0, subtree = NULL, ...)
```

Arguments

x	an object of class C5.0
trial	an integer for how many boosting iterations are used for prediction. NOTE: the internals of C5.0 are zero-based so to get the initial decision tree you must use trial = 0. If trial is set too large, it is reset to the largest value and a warning is given.
subtree	an optional integer that can be used to isolate nodes below the specified split. See partykit::party() for more details.
...	options passed to partykit::plot.party()

Value

No value is returned; a plot is rendered.

Author(s)

Mark Culp, Max Kuhn

References

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

See Also

C5.0(), partykit::party()

Examples

```
mod1 <- C5.0(Species ~ ., data = iris)
plot(mod1)
plot(mod1, subtree = 3)


mod2 <- C5.0(Species ~ ., data = iris, trials = 10)
```

```
plot(mod2) ## should be the same as above

## plot first weighted tree
plot(mod2, trial = 1)
```

`predict.C5.0` *Predict new samples using a C5.0 model*

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 `C5.0`
`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 my 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. #' Factor or character values in `newdata` that were not in the training data are treated as missing, with a warning, and C5.0 handles them as it handles any missing value. (Up to version 0.2.0, they made `predict()` stop with an error when `C5.0()` was called with `x` and `y`.) With the formula method, `stats::model.frame()` still gives an error for new levels of factor and character columns.

Models fitted with version 0.2.0 or earlier can still be used for prediction.

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

```
library(modeldata)
data(mlc_churn)

treeModel <- C5.0(x = mlc_churn[1:3333, -20], y = mlc_churn$churn[1:3333])
predict(treeModel, mlc_churn[3334:3350, -20])
predict(treeModel, mlc_churn[3334:3350, -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

```
library(modeldata)
data(mlc_churn)

treeModel <- C5.0(x = mlc_churn[1:3333, -20], y = mlc_churn$churn[1:3333])
summary(treeModel)
```

Index

∗ **models**
C5.0.default, 2
C5.0Control, 6
C5imp, 7
plot.C5.0, 9
predict.C5.0, 10
summary.C5.0, 12

C5.0 *(*C5.0.default*)*, 2
C5.0(), *7–9*, *11*, *13*
C5.0.default, 2
C5.0Control, 6
C5.0Control(), *2, 3*, *5*, *8*, *10, 11*, *13*
C5imp, 7
C5imp(), *5*, *7*, *11*, *13*

partykit::party(), *9*
partykit::plot.party(), *9*
plot.C5.0, 9
predict.C5.0, 10
predict.C5.0(), *4, 5*, *7, 8*

stats::model.frame(), *11*
summary.C5.0, 12
summary.C5.0(), *5*, *7, 8*, *11*, *13*