Package: OneR (via r-universe)

October 22, 2026

Type Package

Title One Rule Machine Learning Classification Algorithm with Enhancements

Version 2.2

Date 2017-05-05

Author Holger von Jouanne-Diedrich

Maintainer Holger von Jouanne-Diedrich

<holger.jouanne-diedrich@h-ab.de>

Depends R (>= 2.10)

Description Implements the One Rule (OneR) Machine Learning classification algorithm (Holte, R.C. (1993) <doi:10.1023/A:1022631118932>) with enhancements for sophisticated handling of numeric data and missing values together with extensive diagnostic functions. It is useful as a baseline for machine learning models and the rules are often helpful heuristics.

License MIT + file LICENSE

URL https://github.com/vonjd/OneR

BugReports https://github.com/vonjd/OneR/issues

LazyData TRUE

RoxygenNote 6.0.1

Suggests knitr, rmarkdown

VignetteBuilder knitr

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

Date/Publication 2018-06-04 17:09:36 UTC

RemoteUrl https://github.com/vonjd/oner

RemoteRef HEAD

RemoteSha 72dd03301b552d928ddc6bccf3d246c01918f17d

Contents

bin . . . 2
breastcancer . . . 3
eval_model . . . 4
is.OneR . . . 6
maxlevels . . . 6
OneR . . . 7
optbin . . . 9
plot.OneR . . . 11
predict.OneR . . . 12
print.OneR . . . 13
summary.OneR . . . 14

Index **15**

`bin` *Binning function*

Description

Discretizes all numerical data in a data frame into categorical bins of equal length or content or based on automatically determined clusters.

Usage

```
bin(data, nbins = 5, labels = NULL, method = c("length", "content",
  "clusters"), na.omit = TRUE)
```

Arguments

`data`	data frame or vector which contains the data.
`nbins`	number of bins (= levels).
`labels`	character vector of labels for the resulting category.
`method`	character string specifying the binning method, see 'Details'; can be abbreviated.
`na.omit`	logical value whether instances with missing values should be removed.

Details

Character strings and logical strings are coerced into factors. Matrices are coerced into data frames. When called with a single vector only the respective factor (and not a data frame) is returned. Method `"length"` gives intervals of equal length, method `"content"` gives intervals of equal content (via quantiles). Method `"clusters"` determins `"nbins"` clusters via 1D kmeans with deterministic seeding of the initial cluster centres (Jenks natural breaks optimization).

When `"na.omit = FALSE"` an additional level `"NA"` is added to each factor with missing values.

Value

A data frame or vector.

Author(s)

Holger von Jouanne-Diedrich

References

https://github.com/vonjd/OneR

See Also

OneR, optbin

Examples

```
data <- iris
str(data)
str(bin(data))
str(bin(data, nbins = 3))
str(bin(data, nbins = 3, labels = c("small", "medium", "large")))

## Difference between methods "length" and "content"
set.seed(1); table(bin(rnorm(900), nbins = 3))
set.seed(1); table(bin(rnorm(900), nbins = 3, method = "content"))

## Method "clusters"
intervals <- paste(levels(bin(faithful$waiting, nbins = 2, method = "cluster")), collapse = " ")
hist(faithful$waiting, main = paste("Intervals:", intervals))
abline(v = c(42.9, 67.5, 96.1), col = "blue")

## Missing values
bin(c(1:10, NA), nbins = 2, na.omit = FALSE) # adds new level "NA"
bin(c(1:10, NA), nbins = 2)              # omits missing values by default (with warning)
```

---

breastcancer *Breast Cancer Wisconsin Original Data Set*

Description

Dataset containing the original Wisconsin breast cancer data.

Usage

```
data(breastcancer)
```

Format

A data frame with 699 instances and 10 attributes. The variables are as follows:

Details

1. Clump Thickness: 1 - 10
2. Uniformity of Cell Size: 1 - 10
3. Uniformity of Cell Shape: 1 - 10
4. Marginal Adhesion: 1 - 10
5. Single Epithelial Cell Size: 1 - 10
6. Bare Nuclei: 1 - 10
7. Bland Chromatin: 1 - 10
8. Normal Nucleoli: 1 - 10
9. Mitoses: 1 - 10
10. Class: benign, malignant

References

The data were obtained from the UCI machine learning repository, see https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Original)

Examples

```
data(breastcancer)
data <- optbin(breastcancer, method = "infogain")
model <- OneR(data, verbose = TRUE)
summary(model)
plot(model)
prediction <- predict(model, data)
eval_model(prediction, data)
```

`eval_model` *Classification Evaluation function*

Description

Function for evaluating a OneR classification model. Prints confusion matrices with prediction vs. actual in absolute and relative numbers. Additionally it gives the accuracy, error rate as well as the error rate reduction versus the base rate accuracy together with a p-value.

Usage

```
eval_model(prediction, actual, dimnames = c("Prediction", "Actual"),
  zero.print = "0")
```

Arguments

`prediction`	vector which contains the predicted values.
`actual`	data frame which contains the actual data. When there is more than one column the last last column is taken. A single vector is allowed too.
`dimnames`	character vector of printed dimnames for the confusion matrices.
`zero.print`	character specifying how zeros should be printed; for sparse confusion matrices, using "." can produce more readable results.

Details

Error rate reduction versus the base rate accuracy is calculated by the following formula:

$$(Accuracy(Prediction) - Accuracy(Baserate))/(1 - Accuracy(Baserate)),$$

giving a number between 0 (no error reduction) and 1 (no error).

In some borderline cases when the model is performing worse than the base rate negative numbers can result. This shows that something is seriously wrong with the model generating this prediction.

The provided p-value gives the probability of obtaining a distribution of predictions like this (or even more unambiguous) under the assumption that the real accuracy is equal to or lower than the base rate accuracy. More technicaly it is derived from a one-sided binomial test with the alternative hypothesis that the prediction's accuracy is bigger than the base rate accuracy. Loosly speaking a low p-value (< 0.05) signifies that the model really is able to give predictions that are better than the base rate.

Value

Invisibly returns a list with the number of correctly classified and total instances and a confusion matrix with the absolute numbers.

Author(s)

Holger von Jouanne-Diedrich

References

https://github.com/vonjd/OneR

Examples

```
data <- iris
model <- OneR(data)
summary(model)
prediction <- predict(model, data)
eval_model(prediction, data)
```

is.OneR *Test OneR model objects*

Description

Test if object is a OneR model.

Usage

```
is.OneR(x)
```

Arguments

x	object to be tested.

Value

a logical whether object is of class "OneR".

Author(s)

Holger von Jouanne-Diedrich

References

https://github.com/vonjd/OneR

Examples

```
model <- OneR(iris)
is.OneR(model) # evaluates to TRUE
```

maxlevels *Remove factors with too many levels*

Description

Removes all columns of a data frame where a factor (or character string) has more than a maximum number of levels.

Usage

```
maxlevels(data, maxlevels = 20, na.omit = TRUE)
```

Arguments

data	data frame which contains the data.
maxlevels	number of maximum factor levels.
na.omit	logical value whether missing values should be treated as a level, defaults to omit missing values before counting.

Details

Often categories that have very many levels are not useful in modelling OneR rules because they result in too many rules and tend to overfit. Examples are IDs or names.

Character strings are treated as factors although they keep their datatype. Numeric data is left untouched. If data contains unused factor levels (e.g. due to subsetting) these are ignored and a warning is given.

Value

A data frame.

Author(s)

Holger von Jouanne-Diedrich

References

https://github.com/vonjd/OneR

See Also

OneR

Examples

```
df <- data.frame(numeric = c(1:26), alphabet = letters)
str(df)
str(maxlevels(df))
```

OneR *One Rule function*

Description

Builds a model according to the One Rule (OneR) machine learning classification algorithm.

Usage

```
OneR(x, ...)

## S3 method for class 'formula'
OneR(formula, data, ties.method = c("first", "chisq"),
  verbose = FALSE, ...)

## S3 method for class 'data.frame'
OneR(x, ties.method = c("first", "chisq"),
  verbose = FALSE, ...)
```

Arguments

x	data frame with the last column containing the target variable.
...	arguments passed to or from other methods.
formula	formula, additionally the argument data is needed.
data	data frame which contains the data, only needed when using the formula interface.
ties.method	character string specifying how ties are treated, see 'Details'; can be abbreviated.
verbose	if TRUE prints rank, names and predictive accuracy of the attributes in decreasing order (with ties.method = "first").

Details

All numerical data is automatically converted into five categorical bins of equal length. Instances with missing values are removed. This is done by internally calling the default version of bin before starting the OneR algorithm. To finetune this behaviour data preprocessing with the bin or optbin functions should be performed. If data contains unused factor levels (e.g. due to subsetting) these are ignored and a warning is given.

When there is more than one attribute with best performance either the first (from left to right) is being chosen (method "first") or the one with the lowest p-value of a chi-squared test (method "chisq").

Value

Returns an object of class "OneR". Internally this is a list consisting of the function call with the specified arguments, the names of the target and feature variables, a list of the rules, the number of correctly classified and total instances and the contingency table of the best predictor vs. the target variable.

Methods (by class)

- formula: method for formulas.
- data.frame: method for data frames.

Author(s)

Holger von Jouanne-Diedrich

References

https://github.com/vonjd/OneR

See Also

bin, optbin, eval_model, maxlevels

Examples

```
data <- optbin(iris)
model <- OneR(data, verbose = TRUE)
summary(model)
plot(model)
prediction <- predict(model, data)
eval_model(prediction, data)

## The same with the formula interface:
data <- optbin(iris)
model <- OneR(Species ~., data = data, verbose = TRUE)
summary(model)
plot(model)
prediction <- predict(model, data)
eval_model(prediction, data)
```

---

optbin *Optimal Binning function*

Description

Discretizes all numerical data in a data frame into categorical bins where the cut points are optimally aligned with the target categories, thereby a factor is returned. When building a OneR model this could result in fewer rules with enhanced accuracy.

Usage

```
optbin(x, ...)

## S3 method for class 'formula'
optbin(formula, data, method = c("logreg", "infogain",
  "naive"), na.omit = TRUE, ...)

## S3 method for class 'data.frame'
optbin(x, method = c("logreg", "infogain", "naive"),
  na.omit = TRUE, ...)
```

Arguments

x	data frame with the last column containing the target variable.
...	arguments passed to or from other methods.
formula	formula, additionally the argument data is needed.
data	data frame which contains the data, only needed when using the formula interface.
method	character string specifying the method for optimal binning, see 'Details'; can be abbreviated.
na.omit	logical value whether instances with missing values should be removed.

Details

The cutpoints are calculated by pairwise logistic regressions (method "logreg"), information gain (method "infogain") or as the means of the expected values of the respective classes ("naive"). The function is likely to give unsatisfactory results when the distributions of the respective classes are not (linearly) separable. Method "naive" should only be used when distributions are (approximately) normal, although in this case "logreg" should give comparable results, so it is the preferable (and therefore default) method.

Method "infogain" is an entropy based method which calculates cut points based on information gain. The idea is that uncertainty is minimized by making the resulting bins as pure as possible. This method is the standard method of many decision tree algorithms.

Character strings and logical strings are coerced into factors. Matrices are coerced into data frames. If the target is numeric it is turned into a factor with the number of levels equal to the number of values. Additionally a warning is given.

When "na.omit = FALSE" an additional level "NA" is added to each factor with missing values. If the target contains unused factor levels (e.g. due to subsetting) these are ignored and a warning is given.

Value

A data frame with the target variable being in the last column.

Methods (by class)

- formula: method for formulas.
- data.frame: method for data frames.

Author(s)

Holger von Jouanne-Diedrich

References

https://github.com/vonjd/OneR

See Also

OneR, bin

Examples

```
data <- iris # without optimal binning
model <- OneR(data, verbose = TRUE)
summary(model)

data_opt <- optbin(iris) # with optimal binning
model_opt <- OneR(data_opt, verbose = TRUE)
summary(model_opt)

## The same with the formula interface:
data_opt <- optbin(Species ~., data = iris)
model_opt <- OneR(data_opt, verbose = TRUE)
summary(model_opt)
```

---

plot.OneR *Plot Diagnostics for an OneR object*

Description

Plots a mosaic plot for the feature attribute and the target of the OneR model.

Usage

```
## S3 method for class 'OneR'
plot(x, ...)
```

Arguments

x	object of class "OneR".
...	further arguments passed to or from other methods.

Details

If more than 20 levels are present for either the feature attribute or the target the function stops with an error.

Author(s)

Holger von Jouanne-Diedrich

References

https://github.com/vonjd/OneR

See Also

OneR

Examples

```
model <- OneR(iris)
plot(model)
```

---

predict.OneR *Predict method for OneR models*

Description

Predict cases or probabilities based on OneR model object.

Usage

```
## S3 method for class 'OneR'
predict(object, newdata, type = c("class", "prob"), ...)
```

Arguments

object	object of class "OneR".
newdata	data frame in which to look for the feature variable with which to predict.
type	character string denoting the type of predicted value returned. Default "class" gives a named vector with the predicted classes, "prob" gives a matrix whose columns are the probability of the first, second, etc. class.
...	further arguments passed to or from other methods.

Details

newdata can have the same format as used for building the model but must at least have the feature variable that is used in the OneR rules. If cases appear that were not present when building the model the predicted case is UNSEEN or NA when "type = prob".

Value

The default is a factor with the predicted classes, if "type = prob" a matrix is returned whose columns are the probability of the first, second, etc. class.

Author(s)

Holger von Jouanne-Diedrich

References

https://github.com/vonjd/OneR

See Also

OneR

Examples

```
model <- OneR(iris)
prediction <- predict(model, iris[1:4])
eval_model(prediction, iris[5])

## type prob
predict(model, data.frame(Petal.Width = seq(0, 3, 0.5)))
predict(model, data.frame(Petal.Width = seq(0, 3, 0.5)), type = "prob")
```

---

print.OneR *Print OneR models*

Description

`print` method for class `OneR`.

Usage

```
## S3 method for class 'OneR'
print(x, ...)
```

Arguments

`x`	object of class `"OneR"`.
`...`	further arguments passed to or from other methods.

Details

Prints the rules and the accuracy of an OneR model.

Author(s)

Holger von Jouanne-Diedrich

References

https://github.com/vonjd/OneR

See Also

OneR

Examples

```
model <- OneR(iris)
print(model)
```

summary.OneR *Summarize OneR models*

Description

summary method for class OneR.

Usage

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

Arguments

object	object of class "OneR".
...	further arguments passed to or from other methods.

Details

Prints the rules of the OneR model, the accuracy, a contingency table of the feature attribute and the target and performs a chi-squared test on this table.

In the contingency table the maximum values in each column are highlighted by adding a '*', thereby representing the rules of the OneR model.

Author(s)

Holger von Jouanne-Diedrich

References

https://github.com/vonjd/OneR

See Also

OneR

Examples

```
model <- OneR(iris)
summary(model)
```

Index

∗ **1R**
  OneR, 7
∗ **Jenks**
  bin, 2
∗ **OneR**
  is.OneR, 6
  OneR, 7
∗ **One**
  OneR, 7
∗ **Rule**
  OneR, 7
∗ **Wisconsin**
  breastcancer, 3
∗ **accuracy**
  eval_model, 4
∗ **binning**
  bin, 2
  optbin, 9
∗ **breaks**
  bin, 2
∗ **breast**
  breastcancer, 3
∗ **cancer**
  breastcancer, 3
∗ **clusters**
  bin, 2
∗ **datasets**
  breastcancer, 3
∗ **data**
  breastcancer, 3
∗ **diagnostics**
  plot.OneR, 11
  summary.OneR, 14
∗ **discretization**
  bin, 2
  optbin, 9
∗ **discretize**
  bin, 2
  optbin, 9
∗ **evaluation**
  eval_model, 4
∗ **model**
  is.OneR, 6

bin, 2, *8*, *9*, *11*
breastcancer, 3

eval_model, 4, *9*

is.OneR, 6

maxlevels, 6, *9*

OneR, *3*, 7, *7*, *11–14*
optbin, *3*, *8*, 9, *9*

plot.OneR, 11
predict.OneR, 12
print.OneR, 13

summary.OneR, 14