

Package ‘rCBA’

January 8, 2018

Title CBA Classifier for R

Version 0.3.0

Author Jaroslav Kuchar [aut, cre], Tomas Kliegr [ctb]

Maintainer Jaroslav Kuchar <jaroslav.kuchar@gmail.com>

URL <https://github.com/jaroslav-kuchar/rCBA>

BugReports <https://github.com/jaroslav-kuchar/rCBA/issues>

Description Provides implementations of a classifier based on the “Classification Based on Associations” (CBA). It can be used for building classification models from association rules. Rules are pruned in the order of precedence given by the sort criteria and a default rule is added. The final classifier labels provided instances. CBA was originally proposed by Liu, B. Hsu, W. and Ma, Y. Integrating Classification and Association Rule Mining. Proceedings KDD-98, New York, 27-31 August. AAAI. pp80-86 (1998, ISBN:1-57735-070-7).

Depends R (>= 3.1.3), rJava, arules

Imports R.utils, TunePareto, methods, stats, utils

License Apache License (== 2.0)

LazyData true

SystemRequirements Java (>= 8)

RoxygenNote 6.0.1

Collate 'init.R' 'build.R' 'buildFPGrowth.R' 'classification.R'
'fpgrowth.R' 'pruning.R' 'utils.R'

NeedsCompilation no

Repository CRAN

Date/Publication 2018-01-08 20:20:23 UTC

R topics documented:

build	2
buildFPGrowth	3

classification	3
fpgrowth	4
frameToRules	5
pruning	6
Index	7

build	<i>Build classifier function (Apriori-based)</i>
-------	--

Description

Automatic build of the classification model using the Apriori algorithm from the arules

Usage

```
build(trainData, className = NA, pruning = TRUE, sa = list())
```

Arguments

trainData	data.frame or transactions from arules with input data
className	column name with the target class - default is the last column
pruning	performing pruning while building the model
sa	simulated annealing setting. Default values: list(temp=100.0, alpha=0.05, tabuRuleLength=5, timeout=10)

Value

list with parameters and model as data.frame with rules

Examples

```
library("rCBA")
data("iris")

output <- rCBA::build(iris,sa = list(alpha=0.5)) # speeding up the cooling
model <- output$model

predictions <- rCBA::classification(iris, model)
table(predictions)
sum(iris$Species==predictions, na.rm=TRUE) / length(predictions)
```

buildFPGrowth	<i>Build classifier function (FP-Growth-based)</i>
---------------	--

Description

Automatic build of the classification model using the FP-Growth algorithm

Usage

```
buildFPGrowth(train, className = NULL)
```

Arguments

train	data.frame or transactions from arules with input data
className	column name with the target class - default is the last column

Value

list with parameters and model as data.frame with rules

Examples

```
library("rCBA")
data("iris")

output <- rCBA::buildFPGrowth(iris[sample(nrow(iris), 50),], "Species")
model <- output$model

predictions <- rCBA::classification(iris, model)
table(predictions)
sum(iris$Species==predictions, na.rm=TRUE) / length(predictions)
```

classification	<i>A classification function</i>
----------------	----------------------------------

Description

A classification function

Usage

```
classification(test, rules)
```

Arguments

test data.frame or transactions from arules with input data
 rules data.frame with rules

Value

vector with classifications

Examples

```
library("arules")
library("rCBA")
data("iris")

train <- sapply(iris, as.factor)
train <- data.frame(train, check.names=FALSE)
txns <- as(train,"transactions")

rules = apriori(txns, parameter=list(support=0.03, confidence=0.03, minlen=2),
  appearance = list(rhs=c("Species=setosa", "Species=versicolor", "Species=virginica"),default="lhs"))
rulesFrame <- as(rules,"data.frame")

predictions <- rCBA::classification(train,rulesFrame)
table(predictions)
sum(train$Species==predictions,na.rm=TRUE)/length(predictions)
```

 fpgrowth

FP-Growth

Description

FP-Growth algorithm - Jiawei Han, Jian Pei, and Yiyen Yin. Mining frequent patterns without candidate generation. SIGMOD Rec. 29, 2 (2000) <doi:10.1145/335191.335372>

Usage

```
fpgrowth(train, support = 0.01, confidence = 1, maxLength = 5,
  consequent = NULL)
```

Arguments

train data.frame or transactions from arules with input data
 support minimum support
 confidence minimum confidence
 maxLength maximum length
 consequent filter consequent - column name with consequent/target class

Examples

```
library("rCBA")
data("iris")

train <- sapply(iris,as.factor)
train <- data.frame(train, check.names=FALSE)
txns <- as(train,"transactions")

rules = rCBA::fpgrowth(txns, support=0.03, confidence=0.03, maxLength=2, consequent="Species")
rulesFrame <- as(rules,"data.frame")

predictions <- rCBA::classification(train,rulesFrame)
table(predictions)
sum(train$Species==predictions,na.rm=TRUE)/length(predictions)

prunedRulesFrame <- rCBA::pruning(train, rulesFrame, method="m2cba")
predictions <- rCBA::classification(train, prunedRulesFrame)
table(predictions)
sum(train$Species==predictions,na.rm=TRUE)/length(predictions)
```

frameToRules

Conversion of data.frame to rules from arules

Description

Conversion of data.frame to rules from arules

Usage

```
frameToRules(model)
```

Arguments

model data.frame with rules

Value

arules rules representation

Examples

```
library("rCBA")
data("iris")

output <- rCBA::buildFPGrowth(iris[sample(nrow(iris), 50),], "Species")
model <- output$model

rules <- rCBA::frameToRules(model)
inspect(rules)
```

pruning

A Pruning function

Description

A Pruning function

Usage

```
pruning(train, rules, method = "m2cba")
```

Arguments

train	trainData data.frame or transactions from arules with input data
rules	data.frame with rules
method	pruning method m2cba(default) m1cbaldcbrcba

Value

data.frame with pruned rules

Examples

```
library("arules")
library("rCBA")
data("iris")

train <- sapply(iris,as.factor)
train <- data.frame(train, check.names=FALSE)
txns <- as(train,"transactions")

rules = apriori(txns, parameter=list(support=0.03, confidence=0.03, minlen=2),
appearance = list(rhs=c("Species=setosa", "Species=versicolor", "Species=virginica"),default="lhs"))
rulesFrame <- as(rules,"data.frame")

print(nrow(rulesFrame))
prunedRulesFrame <- rCBA::pruning(train, rulesFrame, method="m2cba")
print(nrow(prunedRulesFrame))
```

Index

build, [2](#)

buildFPGrowth, [3](#)

classification, [3](#)

fpgrowth, [4](#)

frameToRules, [5](#)

pruning, [6](#)