Package 'ForestDisc'

July 21, 2025

Type Package

Title Forest Discretization

Version 0.1.0

Author Haddouchi Maïssae

Maintainer Haddouchi Maïssae <maissaem7@gmail.com>

Description Supervised, multivariate, and non-parametric discretization algorithm based on tree ensembles learning and moment matching optimization. This version of the algorithm relies on random forest algorithm to learn a large set of split points that conserves the relationship between attributes and the target class, and on moment matching optimization to transform this set into a reduced number of cut points matching as well as possible statistical properties of the initial set of split points. For each attribute to be discretized, the set S of its related split points extracted through random forest is mapped to a reduced set C of cut points of size k. This mapping relies on minimizing, for each continuous attribute to be discretized, the distance between the four first moments of S and the four first moments of C subject to some constraints. This non-linear optimization problem is performed using k values ranging from 2 to 'max_splits', and the best solution returned correspond to the value k which optimum solution is the lowest one over the different realizations. ForestDisc is a generalization of RFDisc discretization method initially proposed by Berrado and Runger (2009) <doi:10.1109/AICCSA.2009.5069327>, and improved by Berrado et al. in 2012 by adopting the idea of moment matching optimization related by Hoyland and Wallace (2001) <doi:10.1287/mnsc.47.2.295.9834>.

License GPL (>= 3)

Encoding UTF-8

LazyData true

Imports randomForest, nloptr, moments, stats

NeedsCompilation no

Repository CRAN

Date/Publication 2020-03-19 13:00:21 UTC

Contents

Extract_cont_splits 2

ForestDisc 3
RF2Selectedtrees 4
Select_cont_splits 5

Index **7**

`Extract_cont_splits` *Internal function: Continuous split extraction from Random Forest*

Description

Extraction of the splits learned by random forest regarding continuous predictors.

Usage

```
Extract_cont_splits(SelectedTREES)
```

Arguments

`SelectedTREES`	The output of the function RF2Selectedtrees()

Value

List with 2 components:

`continuous_var`	Vector of continuous predictors.
`continuous_splits`	Data frame of splits learned by random forest algorithm regarding continuous predictors.

Author(s)

Haddouchi Maïssae

Examples

```
data(iris)
Mydata=iris
id_target=5
set.seed(1234)
X=Mydata[,1:(id_target-1)]
Y=Mydata[,id_target]
ntree=50
RFTREES=RF2Selectedtrees(X,Y,ntree)
RFCONTSPLITS=Extract_cont_splits(RFTREES)
```

ForestDisc *Multivariate discretization for supervised learning using Random Forest and moment matching optimization*

Description

ForestDisc is a supervised, multivariate and non-parametric discretization algorithm based on tree ensembles learning and moment matching optimization. This version of the algorithm relies on random forest algorithm to learn a large set of split points that conserves the relationship between attributes and the target class, and on moment matching optimization to transform this set into a reduced number of cut points matching as well as possible statistical properties of the initial set of split points. For each attribute to be discretized, the set S of its related split points extracted through random forest is mapped to a reduced set C of cut points of size k.

Usage

```
ForestDisc(data,id_target,ntree=50,max_splits=10,opt_meth="NelderMead")
```

Arguments

data	Data frame to be discretized.
id_target	Column id of the target class.
ntree	Number of trees to grow using random forest algorithm in order to learn split points. The default value is 50.
max_splits	Maximum number of cut points to be used for discretizing continuous attributes in the data. Possible values for 'max_splits' range between 2 and 10. Default value = 10.
opt_meth	The non-linear optimization algorithm to use in order to get the optimal set of cut points matching as well as possible the set of split points. The possible values are DIviding RECTangles algorithm "directL", NelderMead Simplex method "NelderMead", Sequential Least-Squares Quadratic Programming "SLSQP". (more details about these non-linear optimization algorithms can be found in the documentation of the "NLopt" library). The default value used is "NelderMead".

Value

List with components:

Data_disc	Discretized data.
cont_variables	Continuous attributes column ids.
Listcutp	List of cut points used to discretize continuous attributes.
cut_points	Data frame summarizing the best solution returned.
opt_results	Data frame summarizing all the solutions returned for different realizations. Each realization is determined by a size of the set of cut points, ranging between 2 and 'max_splits'.

Author(s)

Haddouchi Maïssae

Examples

```
data(iris)
Mydata=iris
id_target=5
set.seed(1234)
Mydata_Disc=ForestDisc(Mydata,id_target)
```

`RF2Selectedtrees` *Internal function: Trees extraction from Random Forest*

Description

Learn decision splits from random forest algorithm. The resulting model consists of a set of trees where each tree is a collection of rules, and each rule is a combination of decision splits (pairs of variable/value(s)) defined from a root node to a terminal node.

Usage

```
RF2Selectedtrees (X,Y,ntree,max_TreeRules = 'default',min_RuleSupport = 'default')
```

Arguments

`X`	Descriptive attributes data frame.
`Y`	Target attribute (A response vector).
`ntree`	Number of trees to grow using Random Forest algorithm.
`max_TreeRules`	The maximum number of rules in each tree. It represents the maximum number of terminal nodes in each tree grown by random forest. The default value is the one set in random forest algorithm.
`min_RuleSupport`	The minimum support related to each rule (defined from a root node to a leef node). The support of a rule represents the size of its terminal node divided by the number of instances in the data. The default value used is the minimum size of terminal node set in random forest algorithm divided by the number of instances in the data.

Value

List with components:

`ntree`	Number of trees.

list	List of 'ntree' matrix where each one corresponds to a tree grown by random forest algorithm. Each matrix consists of six columns and number of rows equal to the number of nodes in the tree. (more details can be found in the documentation of the function 'getTree' from "randomForest" package)
RF	The original call to randomForest algorithm used.
xlevels	vector of lists of size equal to the number of predictors. Each list corresponds to an attribute. In the case of categorical attribute, the categories are returned. In the case of continuous attribute, the distinct splits values performed by random Forest are returned.
continuous_var	Vector of continuous predictors.
categorical_var	Vector of categorical predictors.

Author(s)

Haddouchi Maïssae

Examples

```
data(iris)
Mydata=iris
id_target=5
set.seed(1234)
X=Mydata[,1:(id_target-1)]
Y=Mydata[,id_target]
ntree=50
RFTREES=RF2Selectedtrees(X,Y,ntree)
```

Select_cont_splits *Internal function: Continuous cut points Selection*

Description

Build the optimal set of cut points C for discretization, based on moment matching. The set of split points S extracted through Extract_cont_splits() function is mapped to a reduced set of cut points C.

Usage

```
Select_cont_splits(cont_splits,max_splits,opt_meth)
```

Arguments

cont_splits	Output of the function Extract_cont_splits().
max_splits	Maximum number of cut points allowed. Possible values range between 2 and 10. Default value = 10.

`opt_meth` The non-linear optimization algorithm to use in order to get the optimal set of cut points matching as well as possible the set of split points. The possible values are DIviding RECTangles algorithm "directL", NelderMead Simplex method "NelderMead", Sequential Least-Squares Quadratic Programming "SLSQP". (more details about these non-linear optimization algorithms can be found in the documentation of the "NLopt" library).

Value

List with 2 components:

`All_splits` Data frame of solutions returned for k values ranging from 2 to 'max_splits'.

`Selected_splits`
Data frame of the best solution returned.

Author(s)

Haddouchi Maïssae

Examples

```
data(iris)
Mydata=iris
id_target=5
set.seed(1234)
X=Mydata[,1:(id_target-1)]
Y=Mydata[,id_target]
ntree=50
RFTREES=RF2Selectedtrees(X,Y,ntree)
RFCONTSPLITS=Extract_cont_splits(RFTREES)
RFSELECTCONTSPLITS=Select_cont_splits(cont_splits=RFCONTSPLITS,max_splits=10,opt_meth="NelderMead")
```

Index

∗ **Discretization**
    ForestDisc, 3
    Select_cont_splits, 5
∗ **Optimization**
    ForestDisc, 3
    Select_cont_splits, 5
∗ **bining**
    ForestDisc, 3
∗ **manip**
    ForestDisc, 3
∗ **multivariate**
    ForestDisc, 3
∗ **nonparametric**
    ForestDisc, 3
∗ **random forest**
    ForestDisc, 3
∗ **supervised**
    ForestDisc, 3
∗ **trees**
    ForestDisc, 3
∗ **tree**
    RF2Selectedtrees, 4

Extract_cont_splits, 2

ForestDisc, 3

RF2Selectedtrees, 4

Select_cont_splits, 5