Package ‘CaseBasedReasoning’

May 2, 2023

Type   Package
Title   Case Based Reasoning
Version 0.3
Date 2023-04-29

Description Case-based reasoning is a problem-solving methodology that involves solving a new problem by referring to the solution of a similar problem in a large set of previously solved problems. The key aspect of Case Based Reasoning is to determine the problem that "most closely" matches the new problem at hand. This is achieved by defining a family of distance functions and using these distance functions as parameters for local averaging regression estimates of the final result. The optimal distance function is chosen based on a specific error measure used in regression estimation. This approach allows for efficient problem-solving by leveraging past experiences and adapting solutions from similar cases. The underlying concept is inspired by the work of Dippon J. (2002) <doi:10.1016/S0167-9473(02)00058-0>.


License MIT + file LICENSE
Depends Rcpp, RcppParallel, rms
Imports R6, ranger, survival, ggplot2, cowplot, dplyr, purrr, tidyr, pryr
Suggests testthat, knitr, rmarkdown, RcppArmadillo
LinkingTo Rcpp, RcppArmadillo, RcppParallel
SystemRequirements GNU make
NeedsCompilation yes
ByteCompile yes
VignetteBuilder knitr
Encoding UTF-8
RoxygenNote 7.2.1

Author Simon Mueller [cre, aut],
          PD Dr. Juergen Dippon [ctb]

Maintainer Simon Mueller <simon.mueller@muon-stat.com>
asDistObject

Converts a distance vector into an object of class dist

Description

Converts a distance vector into an object of class dist

Usage

asDistObject(x, n, method)

Arguments

x  data vector
n  length of x
method  method description
call_function

Call a function by character strings using the namespace and custom parameters.

Description
Call a function by character strings using the namespace and custom parameters.

Usage
call_function(func_list)

Arguments
func_list A list with fields func, namespace, and args

CaseBasedReasoning
Case Based Reasoning

Description
A R package for Case Based Reasoning using statistical/ML models.

CBRBase
Root class for common functionality of this package

Description
Root class for common functionality of this package
Root class for common functionality of this package

Public fields
model the statistical model
data training data
model_fit trained object
formula Object of class formula or character describing the model fit
terms terms of the formula
endPoint Target variable
distMat A matrix with distances
orderMat A matrix with the order indices for similar cases search
Methods

Public methods:

- CBRBase$new()
- CBRBase$fit()
- CBRBase$calc_distance_matrix()
- CBRBase$get_similar_cases()
- CBRBase$clone()

Method new(): Initialize object for searching similar cases

Usage:
CBRBase$new(formula, data)

Arguments:
- formula Object of class formula or character describing the model fit
- data

Method fit(): Fit the Model

Usage:
CBRBase$fit()

Arguments:
- x Training data of class data.frame

Method calc_distance_matrix(): Calculates the distance matrix

Usage:
CBRBase$calc_distance_matrix(query = NULL)

Arguments:
- query Query data of class data.frame
- x Training data of class data.frame

Method get_similar_cases(): Extracts similar cases

Usage:
CBRBase$get_similar_cases(query, k = 1, addDistance = T, merge = F)

Arguments:
- query Query data of class data.frame
- k number of similar cases
- addDistance Add distance to result data.frame
- merge Add query data to matched cases data.frame

Method clone(): The objects of this class are cloneable with this method.

Usage:
CBRBase$clone(deep = FALSE)

Arguments:
- deep Whether to make a deep clone.
Description

Cox-Beta Model for Case-Based-Reasoning

Details

Regression beta coefficients obtained from a CPH regression model fitted on the training data are used for building a weighted distance measure between train and test data. Afterwards, we will use these weights for calculating a \((n \times m)\)-distance matrix, where \(n\) is the number of observations in the training data, and \(m\) is the number of observations of the test data. The user can use this distance matrix for further cluster analysis or for extracting for each test observation \(k\) (= 1,...,\(l\)) similar cases from the train data. We use the rms-package for model fitting, variable selection, and checking model assumptions. If the user omits the test data, this functions returns a \(n \times n\)-distance matrix.

Super classes

*CaseBasedReasoning::CBRBase* -> *CaseBasedReasoning::RegressionModel* -> *CoxModel*

Public fields

- `model` the statistical model
- `model_params` rms arguments

Methods

Public methods:

- `CoxModel$check_ph()`
- `CoxModel$clone()`

Method `check_ph()`: Check proportional hazard assumption graphically

Usage:

```r
CoxModel$check_ph()
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```r
CoxModel$clone(deep = FALSE)
```

Arguments:

- `deep` Whether to make a deep clone.
depth_distance

Depth Distance

Description
This function returns for each observation the pairwise sum of edges between the corresponding terminal nodes over each tree in the random forest.

Usage
depth_distance(x, y = NULL, rfObject)

Arguments
- x: A data.frame with the same columns as in the training data of the RandomForest model
- y: A data.frame with the same columns as in the training data of the RandomForest model
- rfObject: ranger object

Examples

```R
require(ranger)
rf <- ranger(Species ~ ., data = iris, num.trees = 5, write.forest = TRUE)
depth_distance(x=iris[, -5], rfObject=rf)
```

distanceRandomForest

Distance calculation based on RandomForest Proximity or Depth

Description
Distance calculation based on RandomForest Proximity or Depth

Usage
distanceRandomForest(
  x,
  y = NULL,
  rfObject,
  method = "Proximity",
  threads = NULL
)
edges_between_terminal_nodes

Arguments

- **x**: a data.frame
- **y**: a second data.frame
- **rfObject**: ranger object
- **method**: distance calculation method. Proximity (Default) or Depth.
- **threads**: number of threads to use

Value

A dist or a matrix object with pairwise distance of observations in x vs y (if not null)

Examples

```r
library(ranger)
# proximity pairwise distances
rf.fit <- ranger(Species ~ ., data = iris, num.trees = 500, write.forest = TRUE)
distanceRandomForest(x = iris[, -5], rfObject = rf.fit, method = "Proximity", threads = 1)

# depth distance for train versus test subset
set.seed(1234L)
learn <- sample(1:150, 100)
test <- (1:150)[-learn]
rf.fit <- ranger(Species ~ ., data = iris[learn, ], num.trees = 500, write.forest = TRUE)
distanceRandomForest(x = iris[learn, -5], y = iris[test, -5], rfObject = rf.fit, method = "Depth")
```

### edges_between_terminal_nodes

**Number of Edges between Terminal Nodes**

**Description**

First two columns are terminal node IDs; If an ID pair do not appear in a tree -1 is inserted

**Usage**

`edges_between_terminal_nodes(rfObject)`

**Arguments**

- **rfObject**: ranger object

**Value**

A matrix object with pairwise terminal node edge length
Examples

```
require(ranger)
rf.fit <- ranger(Species ~ ., data = iris, num.trees = 5, write.forest = TRUE)
edges_between_terminal_nodes(rf.fit)
```

generate_grid  

### Generate Grid

**Description**

Generates a uniform grid over the distribution of the time2event variable, calculates closest point and returns this point for each input time2event element. Memory consumption will increase when performing the randomForest model with many unique time2event values. Therefore, we offer a reduction of the time2event values by choosing closest elements in a grid.

**Usage**

```
genrate_grid(t2e, grid_length = 250)
```

**Arguments**

- **t2e**: numeric vector with time2event values
- **grid_length**: number of grid elements

**Value**

a list with new_t2e and grid_error

LinearModel  

### Linear Regression Model for Case-Based-Reasoning

**Description**

Linear Regression Model for Case-Based-Reasoning

**Super classes**

- `CaseBasedReasoning::CBRBase` -> `CaseBasedReasoning::RegressionModel` -> `LinearModel`

**Public fields**

- `model` the statistical model
LogisticModel

Methods

Public methods:

• LogisticModel$clone()

Method clone(): The objects of this class are cloneable with this method.

Usage:

LogisticModel$clone(deep = FALSE)

Arguments:

deep Whether to make a deep clone.

LogisticRegressionModel for Case-Based-Reasoning

Description

Logistic Regression Model for Case-Based-Reasoning

Logistic Regression Model for Case-Based-Reasoning

Super classes

CaseBasedReasoning::CBRBase -> CaseBasedReasoning::RegressionModel -> LogisticModel

Public fields

model the statistical model

Methods

Public methods:

• LogisticModel$clone()

Method clone(): The objects of this class are cloneable with this method.

Usage:

LogisticModel$clone(deep = FALSE)

Arguments:

deep Whether to make a deep clone.
**proximity_distance**  
*Get proximity matrix of an ranger object*

**Description**
Get proximity matrix of an ranger object

**Usage**

```
proximity_distance(x, y = NULL, rfObject, as_dist = TRUE)
```

**Arguments**

- `x`: a new dataset
- `y`: a second new dataset (Default: NULL)
- `rfObject`: ranger object
- `as_dist`: Bool, return a dist object.

**Value**

a dist or a matrix object with pairwise proximity of observations in x vs y (if not null)

**Examples**

```
require(ranger)
rf <- ranger(Species ~ ., data = iris, num.trees = 5, write.forest = TRUE)
proximity_distance(x = iris[, -5], rfObject = rf)

set.seed(1234L)
learn <- sample(1:150, 100)
test <- (1:150)[-learn]
rf <- ranger(Species ~ ., data = iris[learn, ], num.trees = 500, write.forest = TRUE)
proximity_distance(x = iris[learn, -5], y = iris[test, -5], rfObject = rf)
```

---

**ranger_forests_to_matrix**  
*Forest2Matrix*

**Description**
Transform trees of a ranger-object to a matrix

**Usage**

```
ranger_forests_to_matrix(rfObject)
```
RegressionModel

Arguments

rfObject ranger object

Value

a matrix object with Column 1: tree ID Column 2: node ID Column 3: child node ID 1 Column 4: child node ID 2

Examples

library(ranger)
rf.fit <- ranger(Species ~ ., data = iris, num.trees = 5, write.forest = TRUE)
forest_matrix <- ranger_forests_to_matrix(rf.fit)

RegressionModel  Root class for Regression Models, e.g., CPH, logistic, and linear regression

Description

Root class for Regression Models, e.g., CPH, logistic, and linear regression

Super class

CaseBasedReasoning::CBRBase -> RegressionModel

Public fields

model_params rms arguments
weights  Weights for distance calculation

Methods

Public methods:

• RegressionModel$print()
• RegressionModel$variable_selection()
• RegressionModel$fit()
• RegressionModel$clone()

Method print(): Prints information of the initialized object

Usage:
RegressionModel$print()
**RFModel**

RandomForest Model for Searching Similar Cases

**Description**

RandomForest Model for Searching Similar Cases

**Details**

This class uses the proximity or depth matrix of the RandomForest algorithm as a similarity matrix of training and query observations. By default all cases with at least one missing values are dropped from learning, calculating the distance matrix and searching for similar cases.

**Super class**

CaseBasedReasoning::CBRBase -> RFModel

**Public fields**

- model the statistical model
- model_params model arguments
- dist_method Distance method
Methods

Public methods:
• RFModel$print()
• RFModel$new()
• RFModel$fit()
• RFModel$set_distance_method()
• RFModel$clone()

Method print(): Prints information of the initialized object
Usage:
RFModel$print()

Method new(): Initialize a RandomForest object for searching similar cases.
Usage:
RFModel$new(formula, data, ...)
Arguments:
formula Object of class formula or character describing the model fit.
data Training data of class data.frame
... ranger RandomForest arguments

Method fit(): Fit the RandomForest
Usage:
RFModel$fit()
Arguments:
x Training data of class data.frame

Method set_distance_method(): Set the distance method. Available are Proximity and Depth
Usage:
RFModel$set_distance_method(method = ”Depth”)
Arguments:
method Distance calculation method (default: Proximity)

Method clone(): The objects of this class are cloneable with this method.
Usage:
RFModel$clone(deep = FALSE)
Arguments:
deep Whether to make a deep clone.

References
**terminalNodes**  
*Get the terminal node id of a RandomForest Object*

**Description**

Extracts for each observation and for each tree in the forest the terminal node id. The index of terminal nodes are starting with 1, e.g., the root node has id 1

**Usage**

`terminalNodes(x, rfObject)`

**Arguments**

- `x` a data.frame
- `rfObject` ranger object

**Value**

Matrix with terminal node IDs for all observations in x (rows) and trees (columns)

**Examples**

```r
library(ranger)
rf.fit <- ranger(Species ~ ., data = iris, num.trees = 5, write.forest = TRUE)
dfNodes <- terminalNodes(iris[, -5], rf.fit)
```

---

**weightedDistance**  
*Weighted Distance calculation*

**Description**

Weighted Distance calculation

**Usage**

`weightedDistance(x, y = NULL, weights = NULL)`

**Arguments**

- `x` a new dataset
- `y` a second new dataset
- `weights` a vector of weights
**weightedDistance**

**Value**

a dist or matrix object

**Examples**

```r
require(ranger)
rf <- ranger(Species ~ ., data = iris, num.trees = 5, write.forest = TRUE)
terminalNodes(iris[, -5], rf)
```
Index

* data-preparation
  RegressionModel, 11
* datapreparation
  CBRBase, 3
  asDistObject, 2
  call_function, 3
  CaseBasedReasoning, 3
  CaseBasedReasoning::CBRBase, 5, 8, 9, 11, 12
  CaseBasedReasoning::RegressionModel, 5, 8, 9
  CBRBase, 3
  CoxModel, 5
  depth_distance, 6
  distanceRandomForest, 6
  edges_between_terminal_nodes, 7
  generate_grid, 8
  LinearModel, 8
  LogisticModel, 9
  proximity_distance, 10
  ranger_forests_to_matrix, 10
  RegressionModel, 11
  RFModel, 12
  terminalNodes, 14
  weightedDistance, 14