Package ‘e2tree’

April 12, 2025

Title Explainable Ensemble Trees

Version 0.1.2

Description The Explainable Ensemble Trees 'e2tree' approach has been proposed by Aria et al. (2024) <doi:10.1007/s00180-022-01312-6>. It aims to explain and interpret decision tree ensemble models using a single tree-like structure. 'e2tree' is a new way of explaining an ensemble tree trained through 'randomForest' or 'xgboost' packages.

License MIT + file LICENSE

URL `https://github.com/massimoaria/e2tree`

BugReports `https://github.com/massimoaria/e2tree/issues`

Encoding UTF-8

RoxygenNote 7.3.2

Imports dplyr, doParallel, parallel, foreach, future.apply, ggplot2, Matrix, partitions, purrr, tidyr, randomForest, rpart.plot, Rcpp, RSpectra, ape

LinkingTo Rcpp

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation yes

Author Massimo Aria [aut, cre, cph] (<`https://orcid.org/0000-0002-8517-9411`>), Agostino Gnasso [aut] (<`https://orcid.org/0000-0002-8046-3923`>)

Maintainer Massimo Aria <`aria@unina.it`>

Repository CRAN

Date/Publication 2025-04-12 08:20:02 UTC

Contents

`createDisMatrix` *Dissimilarity matrix*

Description

The function createDisMatrix creates a dissimilarity matrix among observations from an ensemble tree.

Usage

```
createDisMatrix(
  ensemble,
  data,
  label,
  parallel = list(active = FALSE, no_cores = 1),
  verbose = FALSE
)
```

Arguments

`ensemble`	is an ensemble tree object (for the moment ensemble works only with random forest objects)
`data`	is a data frame containing the variables in the model. It is the data frame used for ensemble learning.
`label`	is a character. It indicates the response label.
`parallel`	A list with two elements: `active` (logical) and `no_cores` (integer). If `active = TRUE`, the function performs parallel computation using the number of cores specified in `no_cores`. If `no_cores` is NULL or equal to 0, it defaults to using all available cores minus one. If `active = FALSE`, the function runs on a single core. Default: `list(active = FALSE, no_cores = 1)`.
`verbose`	Logical. If TRUE, the function prints progress messages and other information during execution. If FALSE (the default), messages are suppressed.

Value

A dissimilarity matrix. This is a dissimilarity matrix measuring the discordance between two observations concerning a given classifier of a random forest model.

Examples

```
## Classification
data("iris")

# Create training and validation set:
smp_size <- floor(0.75 * nrow(iris))
train_ind <- sample(seq_len(nrow(iris)), size = smp_size)
training <- iris[train_ind, ]
validation <- iris[-train_ind, ]
response_training <- training[,5]
response_validation <- validation[,5]

# Perform training:
ensemble <- randomForest::randomForest(Species ~ ., data=training,
importance=TRUE, proximity=TRUE)

D <- createDisMatrix(ensemble, data=training,
                     label = "Species",
                     parallel = list(active=FALSE, no_cores = 1))


## Regression
data("mtcars")

# Create training and validation set:
smp_size <- floor(0.75 * nrow(mtcars))
train_ind <- sample(seq_len(nrow(mtcars)), size = smp_size)
training <- mtcars[train_ind, ]
validation <- mtcars[-train_ind, ]
response_training <- training[,1]
response_validation <- validation[,1]

# Perform training
ensemble = randomForest::randomForest(mpg ~ ., data=training, ntree=1000,
importance=TRUE, proximity=TRUE)

D = createDisMatrix(ensemble, data=training,
                        label = "mpg",
                       parallel = list(active=FALSE, no_cores = 1))
```

`e2tree` *Explainable Ensemble Tree*

Description

It creates an explainable tree for Random Forest. Explainable Ensemble Trees (E2Tree) aimed to generate a “new tree” that can explain and represent the relational structure between the response variable and the predictors. This lead to providing a tree structure similar to those obtained for a decision tree exploiting the advantages of a dendrogram-like output.

Usage

```
e2tree(
  formula,
  data,
  D,
  ensemble,
  setting = list(impTotal = 0.1, maxDec = 0.01, n = 2, level = 5)
)
```

Arguments

`formula`	is a formula describing the model to be fitted, with a response but no interaction terms.
`data`	a data frame containing the variables in the model. It is a data frame in which to interpret the variables named in the formula.
`D`	is the dissimilarity matrix. This is a dissimilarity matrix measuring the discordance between two observations concerning a given classifier of a random forest model. The dissimilarity matrix is obtained with the createDisMatrix function.
`ensemble`	is an ensemble tree object (for the moment ensemble works only with random forest objects)
`setting`	is a list containing the set of stopping rules for the tree building procedure.

`impTotal`	The threshold for the impurity in the node
`maxDec`	The threshold for the maximum impurity decrease of the node
`n`	The minimum number of the observations in the node
`level`	The maximum depth of the tree (levels)

Default is `setting=list(impTotal=0.1, maxDec=0.01, n=2, level=5)`.

Value

A e2tree object, which is a list with the following components:

`tree`	A data frame representing the main structure of the tree aimed at explaining and graphically representing the rel
`call`	The matched call
`terms`	A list of terms and attributes
`control`	A list containing the set of stopping rules for the tree building procedure
`varimp`	A list containing a table and a plot for the variable importance. Variable importance refers to a quantitative mea

Examples

```
## Classification:
data(iris)

# Create training and validation set:
smp_size <- floor(0.75 * nrow(iris))
```

```
train_ind <- sample(seq_len(nrow(iris)), size = smp_size)
training <- iris[train_ind, ]
validation <- iris[-train_ind, ]
response_training <- training[,5]
response_validation <- validation[,5]

# Perform training:
ensemble <- randomForest::randomForest(Species ~ ., data=training,
importance=TRUE, proximity=TRUE)

D <- createDisMatrix(ensemble, data=training, label = "Species",
                              parallel = list(active=FALSE, no_cores = 1))

setting=list(impTotal=0.1, maxDec=0.01, n=2, level=5)
tree <- e2tree(Species ~ ., training, D, ensemble, setting)



## Regression
data("mtcars")

# Create training and validation set:
smp_size <- floor(0.75 * nrow(mtcars))
train_ind <- sample(seq_len(nrow(mtcars)), size = smp_size)
training <- mtcars[train_ind, ]
validation <- mtcars[-train_ind, ]
response_training <- training[,1]
response_validation <- validation[,1]

# Perform training
ensemble = randomForest::randomForest(mpg ~ ., data=training, ntree=1000,
importance=TRUE, proximity=TRUE)

D = createDisMatrix(ensemble, data=training, label = "mpg",
                               parallel = list(active=FALSE, no_cores = 1))

setting=list(impTotal=0.1, maxDec=(1*10^-6), n=2, level=5)
tree <- e2tree(mpg ~ ., training, D, ensemble, setting)
```

`eComparison` *Comparison of Heatmaps and Mantel Test*

Description

This function processes heatmaps for visual comparison and performs the Mantel test between a proximity matrix derived from Random Forest outputs and a matrix estimated by E2Tree. Heatmaps are generated for both matrices. The Mantel test quantifies the correlation between the matrices, offering a statistical measure of similarity.

Usage

```
eComparison(data, fit, D, graph = TRUE)
```

Arguments

`data`	a data frame containing the variables in the model. It is the data frame used for ensemble learning.
`fit`	is e2tree object.
`D`	is the dissimilarity matrix. This is a dissimilarity matrix measuring the discordance between two observations concerning a given classifier of a random forest model. The dissimilarity matrix is obtained with the createDisMatrix function.
`graph`	A logical value (default: TRUE). If TRUE, heatmaps of both matrices are generated and displayed.

Value

A list containing three elements:

- `RF HeatMap`: A heatmap plot of the Random Forest-derived proximity matrix.
- `E2Tree HeatMap`: A heatmap plot of the E2Tree-estimated matrix.
- `Mantel Test`: Results of the Mantel test, including the correlation coefficient and significance level.

Examples

```
## Classification:
data(iris)

# Create training and validation set:
smp_size <- floor(0.75 * nrow(iris))
train_ind <- sample(seq_len(nrow(iris)), size = smp_size)
training <- iris[train_ind, ]
validation <- iris[-train_ind, ]
response_training <- training[,5]
response_validation <- validation[,5]

# Perform training:
ensemble <- randomForest::randomForest(Species ~ ., data=training,
importance=TRUE, proximity=TRUE)

D <- createDisMatrix(ensemble, data=training, label = ”Species”,
                          parallel = list(active=FALSE, no_cores = 1))

setting=list(impTotal=0.1, maxDec=0.01, n=2, level=5)
tree <- e2tree(Species ~ ., training, D, ensemble, setting)

eComparison(training, tree, D)
```

```
## Regression
data("mtcars")

# Create training and validation set:
smp_size <- floor(0.75 * nrow(mtcars))
train_ind <- sample(seq_len(nrow(mtcars)), size = smp_size)
training <- mtcars[train_ind, ]
validation <- mtcars[-train_ind, ]
response_training <- training[,1]
response_validation <- validation[,1]

# Perform training
ensemble = randomForest::randomForest(mpg ~ ., data=training, ntree=1000,
importance=TRUE, proximity=TRUE)

D = createDisMatrix(ensemble, data=training, label = "mpg",
                          parallel = list(active=FALSE, no_cores = 1))

setting=list(impTotal=0.1, maxDec=(1*10^-6), n=2, level=5)
tree <- e2tree(mpg ~ ., training, D, ensemble, setting)

eComparison(training, tree, D)
```

`ePredTree` *Predict responses through an explainable RF*

Description

It predicts classification and regression tree responses

Usage

```
ePredTree(fit, data, target = "1")
```

Arguments

`fit`	is a e2tree object
`data`	is a data frame
`target`	is the target value of response in the classification case

Value

an object.

Examples

```
## Classification:
data(iris)

# Create training and validation set:
smp_size <- floor(0.75 * nrow(iris))
train_ind <- sample(seq_len(nrow(iris)), size = smp_size)
training <- iris[train_ind, ]
validation <- iris[-train_ind, ]
response_training <- training[,5]
response_validation <- validation[,5]

# Perform training:
ensemble <- randomForest::randomForest(Species ~ ., data=training,
importance=TRUE, proximity=TRUE)

D <- createDisMatrix(ensemble, data=training, label = "Species",
                             parallel = list(active=FALSE, no_cores = 1))

setting=list(impTotal=0.1, maxDec=0.01, n=2, level=5)
tree <- e2tree(Species ~ ., training, D, ensemble, setting)

ePredTree(tree, validation, target="1")


## Regression
data("mtcars")

# Create training and validation set:
smp_size <- floor(0.75 * nrow(mtcars))
train_ind <- sample(seq_len(nrow(mtcars)), size = smp_size)
training <- mtcars[train_ind, ]
validation <- mtcars[-train_ind, ]
response_training <- training[,1]
response_validation <- validation[,1]

# Perform training
ensemble = randomForest::randomForest(mpg ~ ., data=training, ntree=1000,
importance=TRUE, proximity=TRUE)

D = createDisMatrix(ensemble, data=training, label = "mpg",
                              parallel = list(active=FALSE, no_cores = 1))

setting=list(impTotal=0.1, maxDec=(1*10^-6), n=2, level=5)
tree <- e2tree(mpg ~ ., training, D, ensemble, setting)

ePredTree(tree, validation)
```

`roc` *Roc curve*

Description

Computes and plots the Receiver Operating Characteristic (ROC) curve for a binary classification model, along with the Area Under the Curve (AUC). The ROC curve is a graphical representation of a classifier's performance across all classification thresholds.

Usage

```
roc(response, scores, target = "1")
```

Arguments

`response`	is the response variable vector
`scores`	is the probability vector of the prediction
`target`	is the target response class

Value

an object.

Examples

```
## Classification:
data(iris)

# Create training and validation set:
smp_size <- floor(0.75 * nrow(iris))
train_ind <- sample(seq_len(nrow(iris)), size = smp_size)
training <- iris[train_ind, ]
validation <- iris[-train_ind, ]
response_training <- training[,5]
response_validation <- validation[,5]

# Perform training:
ensemble <- randomForest::randomForest(Species ~ ., data=training,
importance=TRUE, proximity=TRUE)

D <- createDisMatrix(ensemble, data=training, label = "Species",
                            parallel = list(active=FALSE, no_cores = 1))

setting=list(impTotal=0.1, maxDec=0.01, n=2, level=5)
tree <- e2tree(Species ~ ., training, D, ensemble, setting)

pr <- ePredTree(tree, validation, target="setosa")

roc(response_training, scores = pr$score, target = "setosa")
```

`rpart2Tree` *Convert e2tree into an rpart object*

Description

It converts an e2tree output into an rpart object.

Usage

```
rpart2Tree(fit, ensemble)
```

Arguments

`fit`	is e2tree object.
`ensemble`	is an ensemble tree object (for the moment ensemble works only with random forest objects).

Value

An rpart object. It contains the following components:

`frame`	The data frame includes a singular row for each node present in the tree. The row.names within t
`where`	An integer vector that matches the length of observations in the root node. The vector contains th
`call`	The matched call
`terms`	A list of terms and attributes
`control`	A list containing the set of stopping rules for the tree building procedure
`functions`	The summary, print, and text functions are utilized for the specific method required
`variable.importance`	Variable importance refers to a quantitative measure that assesses the contribution of individual v

Examples

```
## Classification:
data(iris)

# Create training and validation set:
smp_size <- floor(0.75 * nrow(iris))
train_ind <- sample(seq_len(nrow(iris)), size = smp_size)
training <- iris[train_ind, ]
validation <- iris[-train_ind, ]
response_training <- training[,5]
response_validation <- validation[,5]
```

```
# Perform training:
ensemble <- randomForest::randomForest(Species ~ ., data=training,
importance=TRUE, proximity=TRUE)

D <- createDisMatrix(ensemble, data=training, label = "Species",
                             parallel = list(active=FALSE, no_cores = 1))

setting=list(impTotal=0.1, maxDec=0.01, n=2, level=5)
tree <- e2tree(Species ~ ., training, D, ensemble, setting)

# Convert e2tree into an rpart object:
expl_plot <- rpart2Tree(tree, ensemble)

# Plot using rpart.plot package:
rpart.plot::rpart.plot(expl_plot)


## Regression
data("mtcars")

# Create training and validation set:
smp_size <- floor(0.75 * nrow(mtcars))
train_ind <- sample(seq_len(nrow(mtcars)), size = smp_size)
training <- mtcars[train_ind, ]
validation <- mtcars[-train_ind, ]
response_training <- training[,1]
response_validation <- validation[,1]

# Perform training
ensemble = randomForest::randomForest(mpg ~ ., data=training, ntree=1000,
importance=TRUE, proximity=TRUE)

D = createDisMatrix(ensemble, data=training, label = "mpg",
                       parallel = list(active=FALSE, no_cores = 1))

setting=list(impTotal=0.1, maxDec=(1*10^-6), n=2, level=5)
tree <- e2tree(mpg ~ ., training, D, ensemble, setting)

# Convert e2tree into an rpart object:
expl_plot <- rpart2Tree(tree, ensemble)

# Plot using rpart.plot package:
rpart.plot::rpart.plot(expl_plot)
```

`vimp` *Variable Importance*

Description

It calculate variable importance of an explainable tree

Usage

```
vimp(fit, data, type = "classification")
```

Arguments

`fit`	is a e2tree object
`data`	is a data frame in which to interpret the variables named in the formula.
`type`	Specify the type. The default is ‘classification’, otherwise ‘regression’.

Value

a data frame containing variable importance metrics.

Examples

```
## Classification:
data(iris)

# Create training and validation set:
smp_size <- floor(0.75 * nrow(iris))
train_ind <- sample(seq_len(nrow(iris)), size = smp_size)
training <- iris[train_ind, ]
validation <- iris[-train_ind, ]
response_training <- training[,5]
response_validation <- validation[,5]

# Perform training:
ensemble <- randomForest::randomForest(Species ~ ., data=training,
importance=TRUE, proximity=TRUE)

D <- createDisMatrix(ensemble, data=training, label = "Species",
                             parallel = list(active=FALSE, no_cores = 1))

setting=list(impTotal=0.1, maxDec=0.01, n=2, level=5)
tree <- e2tree(Species ~ ., training, D, ensemble, setting)

vimp(tree, training, type = "classification")


## Regression
data("mtcars")

# Create training and validation set:
smp_size <- floor(0.75 * nrow(mtcars))
train_ind <- sample(seq_len(nrow(mtcars)), size = smp_size)
training <- mtcars[train_ind, ]
validation <- mtcars[-train_ind, ]
response_training <- training[,1]
response_validation <- validation[,1]

# Perform training
```

```
ensemble = randomForest::randomForest(mpg ~ ., data=training, ntree=1000,
importance=TRUE, proximity=TRUE)

D = createDisMatrix(ensemble, data=training, label = "mpg",
                         parallel = list(active=FALSE, no_cores = 1))

setting=list(impTotal=0.1, maxDec=(1*10^-6), n=2, level=5)
tree <- e2tree(mpg ~ ., training, D, ensemble, setting)

vimp(tree, training, type = "regression")
```

Index