Package 'e2tree'

March 13, 2026

Title Explainable Ensemble Trees

Version 1.0.0

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 7.3.3

Imports ape, dplyr, parallel, future.apply, ggplot2, Matrix, partitions, purrr, tidyr, Rcpp

LazyData true

LinkingTo Rcpp

Suggests doParallel, foreach, htmlwidgets, jsonlite, randomForest, ranger, rpart.plot, RSpectra, testthat (>= 3.0.0), visNetwork

Config/testthat/edition 3

Depends R (>= 3.5)

NeedsCompilation yes

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

Maintainer Massimo Aria <aria@unina.it>

Repository CRAN

Date/Publication 2026-03-13 17:50:02 UTC

Contents

`createDisMatrix` *Dissimilarity matrix - Optimized for Large Datasets*

Description

The function createDisMatrix creates a dissimilarity matrix among observations from an ensemble tree. This optimized version is designed for large datasets (50K-500K observations) with improved memory management and chunking capabilities.

Usage

```
createDisMatrix(
  ensemble,
  data,
  label,
  parallel = list(active = FALSE, no_cores = 1),
  verbose = FALSE,
  chunk_size = NULL,
  memory_limit = NULL,
  use_disk = FALSE,
  temp_dir = tempdir(),
  batch_aggregate = 10
)
```

Arguments

`ensemble`	is an ensemble tree object
`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.
`chunk_size`	Integer. Number of rows to process in each chunk. If NULL, automatically determined based on available memory and dataset size. Default: NULL (auto).
`memory_limit`	Numeric. Maximum memory to use in GB. Default: NULL (no limit).
`use_disk`	Logical. If TRUE and dataset is very large, intermediate results are saved to disk. Default: FALSE.
`temp_dir`	Character. Directory for temporary files if use_disk = TRUE. Default: tempdir().
`batch_aggregate`	Integer. Number of tree results to aggregate at once before adding to main matrix (reduces memory peaks). Default: 10.

Details

This optimized version implements several strategies for handling large datasets:

- **Memory-efficient aggregation**: Results from parallel trees are aggregated in batches to avoid memory peaks
- **Chunking**: For very large matrices, computation can be split into manageable chunks
- **Sparse matrix optimization**: Maintains sparsity throughout computation
- **Automatic garbage collection**: Explicit memory cleanup at critical points
- **Disk-based computation**: Optional saving of intermediate results for datasets exceeding memory capacity

Supported ensemble types for *classification* or *regression* tasks:

- `randomForest`
- `ranger`

Value

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

Examples

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

## "ranger" package
if (requireNamespace("ranger", quietly = TRUE)) {
  ensemble <- ranger::ranger(Species ~ ., data = iris,
    num.trees = 1000, importance = 'impurity')
}

# Compute dissimilarity matrix with optimizations
D <- createDisMatrix(
  ensemble,
  data = training,
  label = "Species",
  parallel = list(active = FALSE, no_cores = 1),
  chunk_size = 10000,  # Process 10K rows at a time
  batch_aggregate = 20, # Aggregate 20 trees at once
  verbose = TRUE
)
```

---

credit *Credit Scoring Dataset*

Description

A dataset containing socio-economic and banking information for 468 bank clients, used to assess creditworthiness. All variables are categorical.

Usage

```
credit
```

Format

A data frame with 468 rows and 12 columns:

Type_of_client Credit evaluation outcome: "Creditworthy" or "Non-Creditworthy".

Client_Age Age class of the client (e.g., "less than 23 years", "from 23 to 35 years", "from 35 to 50 years", "over 50 years").

Family_Situation Marital/family status of the client (e.g., "single", "married", "divorced").

Account_Tenure Length of the client's relationship with the bank (e.g., "1 year or less", "from 2 to 5 years", "plus 12 years").

Salary_Credited_to_Bank_Account Whether the client's salary is credited to the bank account (e.g., "domicile salary", "no domicile salary").

Ammount_of_Savings Client's level of savings (e.g., "no savings", "less than 5 thousand", "from 5 to 30 thousand", "more than 30 thousand").

Customer_Occupation Employment category of the client (e.g., "employee", "self-employed", "retired").

Average_Account_Balance Average balance held in the account (e.g., "from 2 to 5 thousand", "more than 5 thousand").

Average_Account_Turnover Average monthly turnover on the account (e.g., "Less than 10 thousand", "from 10 to 50 thousand", "more than 50 thousand").

Credit_Card_Transaction_Count_Monthly Number of credit card transactions per month (e.g., "less than 40", "from 40 to 100", "more than 100").

Authorized_Overdraft_Limit Whether the client has an authorized overdraft facility ("Authorised" or "forbidden").

Authorized_to_Issue_Bank_Checks Whether the client is authorized to issue bank checks ("Authorised" or "forbidden").

---

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

```
## "ranger" package
if (requireNamespace("ranger", quietly = TRUE)) {
  ensemble <- ranger::ranger(Species ~ ., data = iris,
    num.trees = 1000, importance = 'impurity')
}

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

## "ranger" package
if (requireNamespace("ranger", quietly = TRUE)) {
  ensemble <- ranger::ranger(formula = mpg ~ ., data = training,
    num.trees = 1000, importance = "permutation")
}

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 *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)
```

`eValidation` *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

```
eValidation(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)

eValidation(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)

eValidation(training, tree, D)
```

goi *Goodness of Interpretability (GoI) Index*

Description

Computes the GoI index measuring how well the E2Tree-estimated proximity matrix reconstructs the original ensemble proximity matrix.

Usage

```
goi(O, O_hat, sample = FALSE, seed = NULL)
```

Arguments

O	Proximity matrix from the ensemble model (n x n), values in the interval 0 to 1
O_hat	Proximity matrix estimated by E2Tree (n x n), values in the interval 0 to 1
sample	Logical. If TRUE, randomly shuffles O_hat values for permutation testing. Default is FALSE.
seed	Random seed for reproducibility when sample is TRUE. Default is NULL.

Details

The statistic is defined as:

$$GoI(O, \hat{O}) = \sum_{i<j} \frac{(o_{ij} - \hat{o}_{ij})^2}{\max(o_{ij}, \hat{o}_{ij})}$$

where:

- o_{ij} are the ensemble proximities
- $\hat{o}_{ij}$ are the E2Tree-estimated proximities
- The sum is computed over all unique pairs $i < j$

The statistic uses a normalized squared difference, where each cell's contribution is weighted by the maximum of the two proximity values. This gives more weight to discrepancies in high-proximity regions.

The metric uses only the lower triangle of the matrix (excluding the diagonal) since proximity matrices are symmetric.

Zero values in the ensemble matrix O are treated as missing (NA) and excluded from the computation.

Value

A numeric value where:

- 0 indicates perfect reconstruction (identical matrices)
- Higher values indicate greater discrepancy between matrices

Examples

```
# Example

data(iris)
smp_size <- floor(0.75 * nrow(iris))
set.seed(42)
train_ind <- sample(seq_len(nrow(iris)), size = smp_size)
training <- iris[train_ind, ]

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)

vs <- eValidation(training, tree, D)
O  <- vs$Proximity_matrix_ensemble
O_hat <- vs$Proximity_matrix_e2tree

goi(O, O_hat)
goi_perm(O, O_hat, alternative = "less", seed = 42)
goi_analysis(O, O_hat, n_perm = 9999, seed = 42)

plot(goi_perm(O, O_hat, n_perm = 9999, seed = 42))


# Example with simulated data
n <- 50
O <- matrix(runif(n^2, 0.3, 1), n, n)
O <- (O + t(O)) / 2
diag(O) <- 1
O_hat <- O + matrix(rnorm(n^2, 0, 0.1), n, n)
O_hat <- pmax(pmin(O_hat, 1), 0)
diag(O_hat) <- 1

goi(O, O_hat)
```

`goi_analysis` *Complete GoI Analysis*

Description

Performs complete GoI analysis including point estimate and permutation test with confidence intervals.

Usage

```
goi_analysis(
  O,
  O_hat,
  n_perm = 999,
  conf.level = 0.95,
  graph = FALSE,
  seed = NULL
)
```

Arguments

O	Proximity matrix from the ensemble model (n x n)
O_hat	Proximity matrix estimated by E2Tree (n x n)
n_perm	Number of permutations (default: 999)
conf.level	Confidence level (default: 0.95)
graph	Logical. If TRUE, displays the null distribution plot. Default is FALSE.
seed	Random seed for reproducibility. Default is NULL.

Value

An object of class "goi_analysis" containing:

estimate	Observed GoI value
ci	Permutation-based confidence interval
p.value	Test p-value
z_stat	Standardized Z statistic
perm	Complete goi_perm object for detailed analysis

Examples

```
n <- 50
O <- matrix(runif(n^2, 0.3, 1), n, n)
O <- (O + t(O)) / 2
diag(O) <- 1
O_hat <- O + matrix(rnorm(n^2, 0, 0.1), n, n)
O_hat <- pmax(pmin(O_hat, 1), 0)
diag(O_hat) <- 1

result <- goi_analysis(O, O_hat, n_perm = 199)
```

goi_perm *Permutation Test for GoI*

Description

Performs a permutation test to assess whether the association between the ensemble proximity matrix and the E2Tree reconstruction is significantly greater than expected by chance. Includes computation of confidence intervals based on the null distribution.

Usage

```
goi_perm(
  O,
  O_hat,
  n_perm = 999,
  alternative = c("greater", "less", "two.sided"),
  conf.level = 0.95,
  graph = FALSE,
  seed = NULL,
  .silent = FALSE
)
```

Arguments

O	Proximity matrix from the ensemble model (n x n)
O_hat	Proximity matrix estimated by E2Tree (n x n)
n_perm	Number of permutations (default: 999)
alternative	Type of alternative hypothesis: "greater", "less", "two.sided"
conf.level	Confidence level for intervals (default: 0.95)
graph	Logical. If TRUE, displays the null distribution plot. Default is FALSE.
seed	Random seed for reproducibility. Default is NULL.
.silent	Logical. If TRUE, suppresses automatic printing. Default is FALSE.

Details

Test Logic:
The test evaluates H0: there is no association between the structure of the ensemble proximity matrix and the E2Tree matrix.
Under H0, randomly shuffling the values in O_hat breaks any structural association with O, generating the null distribution.

P-value Calculation:
The +1 correction in numerator and denominator includes the observed statistic in the count (Phipson & Smyth, 2010), ensuring valid p-values in the interval from 1/(B+1) to 1.

Permutation-based Confidence Intervals:
CIs are computed by shifting the null distribution variability onto the observed statistic.

Value

An object of class "goi_perm" containing:

`statistic`	Observed GoI value
`p.value`	Test p-value
`ci`	Permutation-based confidence interval
`alternative`	Type of test performed
`n_perm`	Number of permutations
`null_dist`	Null distribution (vector of permuted GoI values)
`null_mean`	Mean of the null distribution
`null_sd`	Standard deviation of the null distribution
`z_stat`	Standardized Z statistic

Examples

```
n <- 50
O <- matrix(runif(n^2, 0.3, 1), n, n)
O <- (O + t(O)) / 2
diag(O) <- 1
O_hat <- O + matrix(rnorm(n^2, 0, 0.1), n, n)
O_hat <- pmax(pmin(O_hat, 1), 0)
diag(O_hat) <- 1

result <- goi_perm(O, O_hat, n_perm = 199)
```

---

plot.goi_perm *Plot method for Permutation Test results*

Description

Displays the null distribution with the observed statistic and confidence intervals.

Usage

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

Arguments

`x`	A goi_perm object
`...`	Additional arguments passed to hist()

plot_e2tree *Quick E2Tree Plot (Non-Interactive)*

Description

Displays an E2Tree as a static plot using rpart.plot. For interactive exploration, use plot_e2tree_click().

Usage

```
plot_e2tree(fit, ensemble, main = "E2Tree", ...)
```

Arguments

fit	An e2tree object
ensemble	The ensemble model (randomForest or ranger)
main	Plot title
...	Additional arguments passed to rpart.plot

Value

Invisibly returns the rpart object

plot_e2tree_click *Interactive E2Tree Plot for R Graphics Device*

Description

Displays an E2Tree as an interactive plot in the R graphics device. Click on nodes to see detailed information in the console. Right-click or press ESC to exit interactive mode.

Usage

```
plot_e2tree_click(
  fit,
  data,
  ensemble,
  main = "E2Tree - Click on nodes (ESC to exit)",
  ...
)
```

Arguments

`fit`	An e2tree object
`data`	The training data used to build the tree
`ensemble`	The ensemble model (randomForest or ranger)
`main`	Plot title (default: "E2Tree - Click on nodes (ESC to exit)")
`...`	Additional arguments passed to rpart.plot

Details

This function converts the e2tree object to an rpart object and displays it using rpart.plot. You can then click on any node to see:

- Node ID and type (terminal/internal)
- Number of observations
- Prediction and probability/purity
- Decision path to reach the node
- Class distribution (for classification)
- Split rule (for internal nodes)
- Observations in the node (for terminal nodes)

Value

Invisibly returns the rpart object

Examples

```
## Not run:
# After creating an e2tree object
plot_e2tree_click(tree, training, ensemble)

# Click on nodes to explore
# Press ESC or right-click to exit

## End(Not run)
```

`plot_e2tree_vis` *Interactive E2Tree Plot with visNetwork*

Description

Displays an E2Tree as an interactive network plot using visNetwork. Features: drag nodes anywhere, zoom, pan, click for details. Starts with hierarchical layout, then you can freely move nodes.

Usage

```
plot_e2tree_vis(
  fit,
  data,
  ensemble,
  width = "100%",
  height = "100%",
  direction = "UD",
  node_spacing = 200,
  level_separation = 200,
  colors = NULL,
  show_percent = TRUE,
  show_prob = TRUE,
  show_n = TRUE,
  font_size = 14,
  edge_font_size = 12,
  split_label_style = "rpart",
  max_label_length = 50,
  details_on = "hover",
  navigation_buttons = FALSE,
  free_drag = FALSE
)
```

Arguments

fit	An e2tree object
data	The training data used to build the tree
ensemble	The ensemble model (randomForest or ranger)
width	Width of the widget (default: "100%")
height	Height of the widget (default: "100%")
direction	Layout direction: "UD" (top-down), "DU" (bottom-up), "LR" (left-right), "RL" (right-left)
node_spacing	Spacing between nodes at same level (default: 200)
level_separation	Spacing between levels (default: 200)
colors	Named vector of colors for classes, or NULL for auto
show_percent	Show percentage in nodes (default: TRUE)
show_prob	Show class probabilities in nodes (default: TRUE)
show_n	Show observation count in nodes (default: TRUE)
font_size	Font size for node labels (default: 14)
edge_font_size	Font size for edge labels (default: 12)
split_label_style	How to display split information: • "rpart" - Variable name in node, threshold on edges (like rpart.plot)

- "full" - Full split rule on edges (variable + condition)
- "threshold" - Only threshold values on edges (< 47, >= 47)
- "yesno" - Simple yes/no on edges
- "none" - No labels on edges (hover for details)
- "innode" - Full split rule displayed IN the node (above stats)

`max_label_length`
: Maximum characters for edge labels before truncating (default: 50)

`details_on`
: When to show node details:

- "hover" - Show on mouse hover (default, but may cover other nodes)
- "click" - Show only on click (avoids covering highlighted nodes)
- "none" - No tooltips (use for cleaner visualization)

`navigation_buttons`
: Show navigation buttons (default: FALSE)

`free_drag`
: If TRUE, nodes can be dragged in ALL directions (horizontal, vertical, diagonal). If FALSE (default), nodes can only be moved horizontally within their level.

Value

A visNetwork htmlwidget object

Examples

```
## Not run:
# Basic usage - hierarchical layout, horizontal drag only
plot_e2tree_vis(tree, training, ensemble)

# Enable free dragging in all directions
plot_e2tree_vis(tree, training, ensemble, free_drag = TRUE)

# Split rule shown directly in the node
plot_e2tree_vis(tree, training, ensemble, split_label_style = "innode")

## End(Not run)
```

---

print_e2tree_summary *Print E2Tree Summary*

Description

Prints a comprehensive summary of an E2Tree model including all decision rules, variable importance, and node statistics.

Usage

```
print_e2tree_summary(fit, data)
```

Arguments

fit	An e2tree object
data	The training data

---

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

## "ranger" package
if (requireNamespace("ranger", quietly = TRUE)) {
  ensemble <- ranger::ranger(Species ~ ., data = iris,
    num.trees = 1000, importance = 'impurity')
}

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

## "ranger" package
if (requireNamespace("ranger", quietly = TRUE)) {
  ensemble <- ranger::ranger(formula = mpg ~ ., data = training,
    num.trees = 1000, importance = "permutation")
}

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)
```

`save_e2tree_html` *Save E2Tree visNetwork Plot to HTML*

Description

Save E2Tree visNetwork Plot to HTML

Usage

```
save_e2tree_html(vis, file = "e2tree_plot.html", selfcontained = TRUE)
```

Arguments

`vis`	A visNetwork object from plot_e2tree_vis()
`file`	Output file path (should end with .html)
`selfcontained`	Include all dependencies in single file

`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