Package: e2tree (via r-universe)

June 14, 2026

Title Explainable Ensemble Trees

Version 1.2.0

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

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

LazyData true

LinkingTo Rcpp

Suggests doParallel, foreach, htmlwidgets, jsonlite, knitr, partykit, gbm, lightgbm, randomForest, ranger, xgboost, rmarkdown, RSpectra, testthat (>= 3.0.0), visNetwork

VignetteBuilder knitr

Config/testthat/edition 3

Depends R (>= 3.5)

Config/roxygen2/version 8.0.0

NeedsCompilation yes

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

Maintainer Massimo Aria <aria@unina.it>

Repository https://cran.r-universe.dev

Date/Publication 2026-05-15 15:32:34 UTC

RemoteUrl https://github.com/cran/e2tree

RemoteRef HEAD

RemoteSha d42b7b325230b96f7862016b6673d18dca682a91

Contents

as.rpart *Convert an E2Tree Object to rpart Format*

Description

Coerces an `e2tree` object into an `rpart` object, which can then be used with standard rpart methods for printing, plotting (e.g., via `rpart.plot`), and prediction.

Usage

```
as.rpart(x, ...)

## S3 method for class 'e2tree'
as.rpart(x, ensemble, ...)
```

Arguments

`x`	An e2tree object.
`...`	Additional arguments (ignored).
`ensemble`	The ensemble model used to build the E2Tree. Supported classes: `randomForest`, `ranger`, `xgb.Booster`, `lgb.Booster`, `gbm`, `catboost.CatBoost`.

Value

An `rpart` object.

See Also

`as.party.e2tree` for conversion to partykit format.

Examples

```
data(iris)
smp_size <- floor(0.75 * nrow(iris))
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)

rpart_obj <- as.rpart(tree, ensemble)
```

createDisMatrix *Create a Dissimilarity Matrix from an Ensemble Model*

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`
- `xgb.Booster` (xgboost)
- `lgb.Booster` (lightgbm)
- `gbm` (gbm)
- `catboost.CatBoost` (catboost)

Value

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

Interpretation note (RF vs boosting)

For *bagging* ensembles (`randomForest`, `ranger`) the trees are grown independently on bootstrap samples; co-occurrence in the same leaf captures local similarity in the predictor space. For *boosting* ensembles (`xgb.Booster`, `lgb.Booster`, `gbm`, `catboost`) each tree is fit to the residual of the previous ones, so leaf co-occurrence reflects similarity in the *error-correction trajectory* rather than in the final prediction space. The resulting dissimilarity matrices therefore have systematically different scales (typically $\bar{D} \in [0.85, 0.95]$ for bagging vs. $[0.35, 0.70]$ for boosting). The surrogate tree built on top of `D` should be interpreted accordingly.

The returned matrix carries an `ensemble_backend` attribute identifying the backend used, which downstream functions check to detect mismatched (`D`, `ensemble`) pairs.

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").

`e2splits` *Extract Split Information from an E2Tree Model*

Description

Returns the split matrix and categorical split encoding from a fitted E2Tree model.

Usage

```
e2splits(x, ...)

## S3 method for class 'e2tree'
e2splits(x, ...)
```

Arguments

`x`	An e2tree object.
`...`	Additional arguments (ignored).

Value

A list with components:

splits The split information matrix.

csplit The categorical split encoding matrix.

`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 Using an Explainable Ensemble Tree*

Description

Predicts classification and regression tree responses.

Usage

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

Arguments

`fit`	An e2tree object.
`data`	A data frame with new observations.
`target`	Target class for classification scoring.

Details

Deprecated: Use `predict.e2tree` instead.

Value

A data frame with predictions.

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)

## Preferred method:
predict(tree, newdata = validation, target = "1")

## Legacy function (deprecated):
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)

## Preferred method:
predict(tree, newdata = validation)
```

```
## Legacy function (deprecated):
ePredTree(tree, validation)
```

`eValidation` *Validate an E2Tree Model via Proximity Matrix Comparison*

Description

Compares the ensemble proximity matrix with the E2Tree-estimated proximity matrix using multiple divergence and similarity measures. Can perform the Mantel test, permutation tests on divergence/similarity measures (nLoI, Hellinger, wRMSE, RV, SSIM), or both.

Usage

```
eValidation(
  data,
  fit,
  D,
  test = c("both", "mantel", "measures"),
  graph = TRUE,
  n_perm = 999,
  conf.level = 0.95,
  seed = NULL
)
```

Arguments

`data`	A data frame containing the variables in the model.
`fit`	An e2tree object.
`D`	The dissimilarity matrix obtained with `createDisMatrix`.
`test`	Character string specifying which tests to perform. One of `"both"` (default), `"mantel"` (Mantel test only), or `"measures"` (divergence/similarity measures with permutation tests only).
`graph`	Logical (default TRUE). If TRUE, heatmaps are displayed.
`n_perm`	Integer. Number of permutations for the permutation test on measures. Default is 999. Set to 0 to skip permutation testing. Ignored when `test = "mantel"`.
`conf.level`	Numeric. Confidence level for intervals. Default is 0.95.
`seed`	Integer or NULL. Random seed for reproducibility.

Value

An object of class "eValidation" containing:

Proximity_matrix_ensemble Ensemble proximity matrix (reordered)

Proximity_matrix_e2tree E2Tree proximity matrix (reordered)

mantel_test Mantel test result (NULL if test = "measures")

loi LoI object with decomposition (NULL if test = "mantel")

measures Data frame with all measures (NULL if test = "mantel")

permutation Permutation test results for measures (if applicable)

Examples

```
## Classification:
data(iris)
smp_size <- floor(0.75 * nrow(iris))
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)

val <- eValidation(training, tree, D, n_perm = 199)
print(val)
summary(val)
plot(val)
```

---

extract_terminal_nodes

Extract Terminal Node Assignments from an Ensemble Model

Description

Returns a data.frame with n_obs rows and n_trees columns where each cell is the terminal-node index assigned to that observation by that tree.

Usage

```
extract_terminal_nodes(ensemble, data)
```

Arguments

ensemble	A trained ensemble model.
data	A data.frame of observations (may include the response column; it is ignored internally).

Value

A data.frame with n_obs rows and n_trees columns of integer terminal-node identifiers.

---

fitted.e2tree *Extract Fitted Values from an E2Tree Model*

Description

Returns the fitted values (predictions) for the training data used to build the E2Tree model.

Usage

```
## S3 method for class 'e2tree'
fitted(object, ...)
```

Arguments

object	An e2tree object.
...	Additional arguments (ignored).

Value

A vector of fitted values.

Examples

```
data(iris)
smp_size <- floor(0.75 * nrow(iris))
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)

fitted(tree)
```

get_ensemble_predictions *Get Training Predictions from an Ensemble Model*

Description

Returns a numeric vector of length `n_obs` with the ensemble's prediction for every training observation. For models that store out-of-bag (OOB) predictions (`randomForest`, `ranger`) the stored OOB vector is returned; for other models in-sample predictions are computed from the training data.

Usage

```
get_ensemble_predictions(ensemble, data, type)
```

Arguments

`ensemble`	A trained ensemble model.
`data`	The training `data.frame` that was used to fit the model.
`type`	Character: `"classification"` or `"regression"`.

Value

Numeric vector of length `nrow(data)`.

get_ensemble_type *Determine Task Type from a Trained Ensemble Model*

Description

Returns `"classification"` or `"regression"` depending on the objective used to train the ensemble.

Usage

```
get_ensemble_type(ensemble)
```

Arguments

`ensemble`	A trained ensemble model. Supported classes: `randomForest`, `ranger`, `xgb.Booster`, `lgb.Booster`, `gbm`, `catboost.CatBoost`.

Value

Character scalar: `"classification"` or `"regression"`.

`loi` *Loss of Interpretability (LoI) Index*

Description

Computes the LoI index and its decomposition, measuring how well the E2Tree-estimated proximity matrix reconstructs the original ensemble proximity matrix.

Usage

```
loi(O, O_hat, normalize = TRUE)
```

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
`normalize`	Logical. If TRUE (default), returns nLoI (divided by M). If FALSE, returns raw LoI.

Details

The statistic is defined as:

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

The Normalized LoI divides by the number of pairs $M = n(n-1)/2$:

$$\mathrm{nLoI}(O, \hat{O}) = \frac{1}{M}\mathrm{LoI}(O, \hat{O})$$

The LoI decomposes into two components:

- **LoI_in**: within-node loss (pairs grouped together by E2Tree)
- **LoI_out**: between-node loss (pairs separated by E2Tree)

The per-pair averages `mean_in` and `mean_out` enable direct comparison between the two components despite their different pair counts.

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.

Decomposition interpretation (per-pair averages):

- `mean_out`: average ensemble proximity lost by the partition. Low values (< 0.1) indicate the tree correctly separates low-proximity pairs. High values (> 0.3) suggest the tree splits apart pairs that the ensemble considers similar –more terminal nodes may help.
- `mean_in`: average calibration error within nodes. Low values (< 0.01) indicate excellent within-node reconstruction. Higher values reflect the inherent fuzzy-to-crisp transition.

Value

An object of class `"loi"` containing:

`loi`	Raw LoI value (unnormalized)
`nloi`	Normalized LoI (LoI / M)
`loi_in`	Within-node component (total)
`loi_out`	Between-node component (total)
`mean_in`	Per-pair average within-node loss (comparable with mean_out)
`mean_out`	Per-pair average between-node loss (comparable with mean_in)
`n`	Matrix dimension
`m`	Number of unique pairs
`n_within`	Number of within-node pairs
`n_between`	Number of between-node pairs

Examples

```
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)
prox <- proximity(vs)
O <- prox$ensemble
O_hat <- prox$e2tree

# Compute LoI with decomposition
result <- loi(O, O_hat)
print(result)
summary(result)
plot(result)

# Permutation test
perm <- loi_perm(O, O_hat, n_perm = 999, seed = 42)
print(perm)
plot(perm)
```

`loi_perm` *Permutation Test for LoI*

Description

Performs a permutation test using row/column permutation to assess whether the E2Tree reconstruction is significantly better than expected by chance.

Usage

```
loi_perm(O, O_hat, n_perm = 999, conf.level = 0.95, 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 for intervals (default: 0.95)
`seed`	Random seed for reproducibility. Default is NULL.

Details

The test uses **simultaneous row/column permutation** of $\hat{O}$: for each replicate, a random permutation π of $\{1, \ldots, n\}$ is drawn and $\hat{O}^{\pi} = \hat{O}[\pi, \pi]$ is computed. This preserves the block-diagonal structure of $\hat{O}$ while breaking the correspondence with O.

The null hypothesis is: the E2Tree labeling is unrelated to the ensemble structure. Under H1 (good reconstruction), the observed nLoI should be significantly *lower* than the null distribution.

P-values include the +1 correction of Phipson & Smyth (2010).

Value

An object of class `"loi_perm"` containing:

`observed`	Observed nLoI value and decomposition (loi object)
`statistic`	Observed nLoI value (scalar)
`p.value`	Test p-value (one-sided, less)
`ci`	Permutation-based confidence interval for nLoI
`null_dist`	Null distribution of nLoI values
`null_mean`	Mean of the null distribution
`null_sd`	Standard deviation of the null distribution
`z_stat`	Standardized Z statistic
`n_perm`	Number of permutations
`conf.level`	Confidence level

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, 0.05), n, n)
O_hat <- pmin(pmax((O_hat + t(O_hat)) / 2, 0), 1); diag(O_hat) <- 1

result <- loi_perm(O, O_hat, n_perm = 199, seed = 42)
print(result)
summary(result)
plot(result)
```

`measures` *Extract Validation Measures*

Description

Extracts the data frame of validation measures from an eValidation object, including divergence and similarity metrics between the ensemble and E2Tree proximity matrices.

Usage

```
measures(x, ...)

## S3 method for class 'eValidation'
measures(x, ...)
```

Arguments

`x`	An eValidation object.
`...`	Additional arguments (ignored).

Value

A data frame with columns for method name, type, observed value, and (if permutation tests were performed) null distribution statistics and p-values.

nodes *Extract Tree Node Information*

Description

Extracts the data frame describing the nodes of an E2Tree model, including split rules, predictions, and node statistics.

Usage

```
nodes(x, ...)

## S3 method for class 'e2tree'
nodes(x, terminal = FALSE, ...)
```

Arguments

x	An e2tree object.
...	Additional arguments (ignored).
terminal	Logical. If TRUE, return only terminal (leaf) nodes. Default is FALSE.

Value

A data frame with one row per node.

Examples

```
data(iris)
smp_size <- floor(0.75 * nrow(iris))
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)

nodes(tree)
nodes(tree, terminal = TRUE)
```

plot.e2tree *Plot an E2Tree model*

Description

Displays the tree structure using rpart.plot. This is a convenience wrapper around `plot_e2tree`.

Usage

```
## S3 method for class 'e2tree'
plot(x, ensemble = NULL, main = "E2Tree", ...)
```

Arguments

`x`	An e2tree object
`ensemble`	The ensemble model (randomForest or ranger). Required for converting the tree to rpart format. Supported classes: `randomForest`, `ranger`, `xgb.Booster`, `lgb.Booster`, `gbm`, `catboost.CatBoost`.
`main`	Plot title. Default is "E2Tree".
`...`	Additional arguments passed to `rpart.plot::rpart.plot`

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

```
# After creating an e2tree object (requires interactive session)
if (interactive()) {
  plot_e2tree_click(tree, training, ensemble)
}
```

`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

```
data(iris)
set.seed(42)
smp_size <- floor(0.75 * nrow(iris))
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)

# Basic usage
plot_e2tree_vis(tree, training, ensemble)
```

`predict.e2tree` *Predict Responses from an E2Tree Model*

Description

Predicts classification or regression responses for new data using the fitted E2Tree model.

Usage

```
## S3 method for class 'e2tree'
predict(object, newdata, target = NULL, ...)
```

Arguments

`object`	An e2tree object.
`newdata`	A data frame containing the new observations. If missing, the fitted values for the training data are returned.
`target`	Character string specifying the target class for computing classification scores. Only used for classification trees. Default is `NULL`, which uses the first level.
`...`	Additional arguments (ignored).

Value

For regression: a data frame with columns `fit` (predicted value) and `sd` (standard deviation of the response within the terminal node, computed from the training data). For classification: a data frame with columns `fit` (predicted class), `accuracy` (probability of the predicted class), and `score` (probability of the target class).

Examples

```
data(iris)
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, ]
```

```
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)

## Predict on new data
pred <- predict(tree, newdata = validation)
```

`print.e2tree` *Print an E2Tree model*

Description

Displays a compact summary of the fitted E2Tree model including task type, tree size, terminal nodes, and splitting variables.

Usage

```
## S3 method for class 'e2tree'
print(x, ...)
```

Arguments

`x`	An e2tree object
`...`	Additional arguments (ignored)

`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

proximity *Extract Proximity Matrices*

Description

Extracts proximity matrices from an eValidation object. The ensemble proximity matrix is derived from the original ensemble model, while the E2Tree proximity matrix is estimated from the fitted E2Tree.

Usage

```
proximity(x, ...)

## S3 method for class 'eValidation'
proximity(x, type = c("both", "ensemble", "e2tree"), ...)
```

Arguments

x	An eValidation object.
...	Additional arguments (ignored).
type	Character string specifying which proximity matrix to extract. One of "ensemble", "e2tree", or "both" (default).

Value

A matrix (if type is "ensemble" or "e2tree") or a list of two matrices (if type is "both").

residuals.e2tree *Extract Residuals from an E2Tree Model*

Description

Returns the residuals (observed minus fitted) for regression E2Tree models. Not available for classification models.

Usage

```
## S3 method for class 'e2tree'
residuals(object, ...)
```

Arguments

object	An e2tree object.
...	Additional arguments (ignored).

Value

A numeric vector of residuals.

Examples

```
data("mtcars")
smp_size <- floor(0.75 * nrow(mtcars))
train_ind <- sample(seq_len(nrow(mtcars)), size = smp_size)
training <- mtcars[train_ind, ]

ensemble <- randomForest::randomForest(mpg ~ ., data = training, ntree = 500,
  importance = TRUE, proximity = TRUE)
D <- createDisMatrix(ensemble, data = training, label = "mpg",
  parallel = list(active = FALSE, no_cores = 1))
setting <- list(impTotal = 0.1, maxDec = 1e-6, n = 2, level = 5)
tree <- e2tree(mpg ~ ., training, D, ensemble, setting)

residuals(tree)
```

---

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 an E2Tree Object to rpart Format*

Description

Converts an e2tree output into an rpart object.

Usage

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

Arguments

`fit`	is e2tree object.
`ensemble`	A trained ensemble model. Supported classes: `randomForest`, `ranger`, `xgb.Booster`, `lgb.Booster`, `gbm`, `catboost.CatBoost`.

Details

Note: `as.rpart.e2tree` is the preferred coercion method. This function is kept for backward compatibility.

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)

## Preferred coercion method:
rpart_obj <- as.rpart(tree, ensemble)

## Legacy function (see as.rpart):
rpart_obj <- rpart2Tree(tree, ensemble)

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

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

summary.e2tree *Summary of an E2Tree model*

Description

Displays a comprehensive summary including tree structure, decision rules, terminal node statistics, and variable importance.

Usage

```
## S3 method for class 'e2tree'
summary(object, ...)
```

Arguments

object	An e2tree object
...	Additional arguments (ignored)

vimp *Variable Importance*

Description

Computes variable importance for an E2Tree model based on mean impurity decrease and (for classification) mean accuracy decrease.

Usage

```
vimp(fit, data, type = NULL)
```

Arguments

fit	An e2tree object.
data	A data frame containing the variables in the model.
type	Character string: "classification" or "regression". If NULL (default), the type is automatically detected from the e2tree object.

Value

A list containing:

vimp A data frame with variable importance metrics.

g_imp A ggplot bar chart of Mean Impurity Decrease.

g_acc (Classification only) A ggplot bar chart of Mean Accuracy Decrease.

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, ]

# 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)

vi <- vimp(tree, training)
```

```
vi$vimp
vi$g_imp


## 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, ]

# 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)

vi <- vimp(tree, training)
vi$vimp
vi$g_imp
```

Index