

Package: ocf (via r-universe)

September 5, 2024

Type Package

Title Ordered Correlation Forest

Version 1.0.1

Description Machine learning estimator specifically optimized for handling ordered non-numeric outcomes. 'ocf' provides forest-based estimation of the conditional choice probabilities and the covariates' marginal effects. Under an ``honesty'' condition, the estimates are consistent and asymptotically normal and standard errors can be obtained by leveraging the weight-based representation of the random forest predictions. Please reference the use as Di Francesco (2023) <[arXiv:2309.08755](https://arxiv.org/abs/2309.08755)>.

License GPL-3

Encoding UTF-8

Depends R (>= 3.4.0)

Imports Rcpp, Matrix, stats, utils, stringr, orf, glmnet, ranger

LinkingTo Rcpp, RcppEigen

RoxygenNote 7.2.3

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

URL <https://riccardo-df.github.io/ocf/>,
<https://github.com/riccardo-df/ocf>

BugReports <https://github.com/riccardo-df/ocf/issues>

Biarch TRUE

Repository <https://riccardo-df.r-universe.dev>

RemoteUrl <https://github.com/riccardo-df/ocf>

RemoteRef HEAD

RemoteSha 70761f6e34fed468379776b41d723479e037f8f2

Contents

generate_ordered_data	2
marginal_effects	4
mean_squared_error	5
multinomial_ml	8
ocf	9
ordered_ml	12
predict.mml	13
predict.ocf	15
predict.oml	16
print.ocf	18
print.ocf.marginal	19
summary.ocf	20
summary.ocf.marginal	21
tree_info	23
Index	25

generate_ordered_data *Generate Ordered Data*

Description

Generate a synthetic data set with an ordered non-numeric outcome, together with conditional probabilities and covariates' marginal effects.

Usage

generate_ordered_data(n)

Arguments

n Sample size.

Details

First, a latent outcome is generated as follows:

$$Y_i^* = g(X_i) + \epsilon_i$$

with:

$$g(X_i) = X_i^T \beta$$

$$X_i := (X_{i,1}, X_{i,2}, X_{i,3}, X_{i,4}, X_{i,5}, X_{i,6})$$

$$X_{i,1}, X_{i,3}, X_{i,5} \sim \mathcal{N}(0, 1)$$

$$X_{i,2}, X_{i,4}, X_{i,6} \sim \text{Bernoulli}(0, 1)$$

$$\beta = (1, 1, 1/2, 1/2, 0, 0)$$

$$\epsilon_i \sim \text{logistic}(0, 1)$$

Second, the observed outcomes are obtained by discretizing the latent outcome into three classes using uniformly spaced threshold parameters.

Third, the conditional probabilities and the covariates' marginal effects at the mean are generated using standard textbook formulas. Marginal effects are approximated using a sample of 1,000,000 observations.

Value

A list storing a data frame with the observed data, a matrix of true conditional probabilities, and a matrix of true marginal effects at the mean of the covariates.

Author(s)

Riccardo Di Francesco

References

- Di Francesco, R. (2023). Ordered Correlation Forest. arXiv preprint [arXiv:2309.08755](https://arxiv.org/abs/2309.08755).

See Also

[ocf](#)

Examples

```
## Generate synthetic data.
set.seed(1986)

data <- generate_ordered_data(1000)

head(data$true_probs)
data$me_at_mean

sample <- data$sample
Y <- sample$Y
X <- sample[, -1]

## Fit ocf.
forests <- ocf(Y, X)
```

marginal_effects *Marginal Effects for Ordered Correlation Forest*

Description

Nonparametric estimation of marginal effects using an [ocf](#) object.

Usage

```
marginal_effects(
  object,
  data = NULL,
  which_covariates = c(),
  eval = "atmean",
  bandwidth = 0.1,
  inference = FALSE
)
```

Arguments

object	An ocf object.
data	Data set of class <code>data.frame</code> to estimate marginal effects. It must contain at least the same covariates used to train the forests. If <code>NULL</code> , marginal effects are estimated on <code>object\$full_data</code> .
which_covariates	Character vector storing the names of the covariates for which marginal effect estimation is desired. If empty (the default), marginal effects are estimated for all covariates.
eval	Evaluation point for marginal effects. Either "mean", "atmean" or "atmedian".
bandwidth	How many standard deviations <code>x_up</code> and <code>x_down</code> differ from <code>x</code> .
inference	Whether to extract weights and compute standard errors. The weights extraction considerably slows down the program.

Details

[marginal_effects](#) can estimate mean marginal effects, marginal effects at the mean, or marginal effects at the median, according to the `eval` argument.

The routine assumes that covariates with more than ten unique values are continuous. Otherwise, covariates are assumed to be discrete.

Value

Object of class `ocf.marginal`.

Author(s)

Riccardo Di Francesco

References

- Di Francesco, R. (2023). Ordered Correlation Forest. arXiv preprint [arXiv:2309.08755](https://arxiv.org/abs/2309.08755).

See Also

[ocf](#)

Examples

```
\donttest{## Generate synthetic data.
set.seed(1986)

data <- generate_ordered_data(1000)
sample <- data$sample
Y <- sample$Y
X <- sample[, -1]

## Fit ocf.
forests <- ocf(Y, X)

## Marginal effects at the mean.
me <- marginal_effects(forests, eval = "atmean")

print(me)
print(me, latex = TRUE)

## Compute standard errors. This requires honest forests.
honest_forests <- ocf(Y, X, honesty = TRUE)

honest_me <- marginal_effects(honest_forests, eval = "atmean", inference = TRUE)

print(honest_me, latex = TRUE)}
```

mean_squared_error

Accuracy Measures for Ordered Probability Predictions

Description

Accuracy measures for evaluating ordered probability predictions.

Usage

```
mean_squared_error(y, predictions, use.true = FALSE)
```

```
mean_absolute_error(y, predictions, use.true = FALSE)
```

```
mean_ranked_score(y, predictions, use.true = FALSE)
```

```
classification_error(y, predictions)
```

Arguments

`y` Either the observed outcome vector or a matrix of true probabilities.

`predictions` Predictions.

`use.true` If TRUE, then the program treats `y` as a matrix of true probabilities.

Details**MSE, MAE, and RPS:**

When calling one of `mean_squared_error`, `mean_absolute_error`, or `mean_ranked_score`, `predictions` must be a matrix of predicted class probabilities, with as many rows as observations in `y` and as many columns as classes of `y`.

If `use.true == FALSE`, the mean squared error (MSE), the mean absolute error (MAE), and the mean ranked probability score (RPS) are computed as follows:

$$MSE = \frac{1}{n} \sum_{i=1}^n \sum_{m=1}^M (1(Y_i = m) - \hat{p}_m(x))^2$$

$$MAE = \frac{1}{n} \sum_{i=1}^n \sum_{m=1}^M |1(Y_i = m) - \hat{p}_m(x)|$$

$$RPS = \frac{1}{n} \sum_{i=1}^n \frac{1}{M-1} \sum_{m=1}^M (1(Y_i \leq m) - \hat{p}_m^*(x))^2$$

If `use.true == TRUE`, the MSE, the MAE, and the RPS are computed as follows (useful for simulation studies):

$$MSE = \frac{1}{n} \sum_{i=1}^n \sum_{m=1}^M (p_m(x) - \hat{p}_m(x))^2$$

$$MAE = \frac{1}{n} \sum_{i=1}^n \sum_{m=1}^M |p_m(x) - \hat{p}_m(x)|$$

$$RPS = \frac{1}{n} \sum_{i=1}^n \frac{1}{M-1} \sum_{m=1}^M (p_m^*(x) - \hat{p}_m^*(x))^2$$

where:

$$p_m(x) = P(Y_i = m | X_i = x)$$

$$p_m^*(x) = P(Y_i \leq m | X_i = x)$$

Classification error:

When calling `classification_error`, predictions must be a vector of predicted class labels.

Classification error (CE) is computed as follows:

$$CE = \frac{1}{n} \sum_{i=1}^n 1(Y_i \neq \hat{Y}_i)$$

where Y_i are the observed class labels.

Value

The MSE, the MAE, the RPS, or the CE of the method.

Author(s)

Riccardo Di Francesco

See Also

[mean_ranked_score](#)

Examples

```
## Generate synthetic data.
set.seed(1986)

data <- generate_ordered_data(1000)
sample <- data$sample
Y <- sample$Y
X <- sample[, -1]

## Training-test split.
train_idx <- sample(seq_len(length(Y)), floor(length(Y) * 0.5))

Y_tr <- Y[train_idx]
X_tr <- X[train_idx, ]

Y_test <- Y[-train_idx]
X_test <- X[-train_idx, ]

## Fit ocf on training sample.
forests <- ocf(Y_tr, X_tr)
```

```
## Accuracy measures on test sample.
predictions <- predict(forests, X_test)

mean_squared_error(Y_test, predictions$probabilities)
mean_ranked_score(Y_test, predictions$probabilities)
classification_error(Y_test, predictions$classification)
```

multinomial_ml *Multinomial Machine Learning*

Description

Estimation strategy to estimate conditional choice probabilities for ordered non-numeric outcomes.

Usage

```
multinomial_ml(y = NULL, X = NULL, learner = "forest", scale = TRUE)
```

Arguments

y	Outcome vector.
X	Covariate matrix (no intercept).
learner	String, either "forest" or "l1". Selects the base learner to estimate each expectation.
scale	Logical, whether to scale the covariates. Ignored if learner is not "l1".

Details

Multinomial machine learning expresses conditional choice probabilities as expectations of binary variables:

$$p_m(X_i) = \mathbb{E}[1(Y_i = m) | X_i]$$

This allows us to estimate each expectation separately using any regression algorithm to get an estimate of conditional probabilities.

`multinomial_ml` combines this strategy with either regression forests or penalized logistic regression with an L1 penalty, according to the user-specified parameter learner.

If learner == "l1", the penalty parameters are chosen via 10-fold cross-validation and `model.matrix` is used to handle non-numeric covariates. Additionally, if scale == TRUE, the covariates are scaled to have zero mean and unit variance.

Value

Object of class `mml`.

Author(s)

Riccardo Di Francesco

References

- Di Francesco, R. (2023). Ordered Correlation Forest. arXiv preprint [arXiv:2309.08755](https://arxiv.org/abs/2309.08755).

See Also

[ordered_ml](#), [ocf](#)

Examples

```
## Generate synthetic data.
set.seed(1986)

data <- generate_ordered_data(1000)
sample <- data$sample
Y <- sample$Y
X <- sample[, -1]

## Training-test split.
train_idx <- sample(seq_len(length(Y)), floor(length(Y) * 0.5))

Y_tr <- Y[train_idx]
X_tr <- X[train_idx, ]

Y_test <- Y[-train_idx]
X_test <- X[-train_idx, ]

## Fit multinomial machine learning on training sample using two different learners.
multinomial_forest <- multinomial_ml(Y_tr, X_tr, learner = "forest")
multinomial_l1 <- multinomial_ml(Y_tr, X_tr, learner = "l1")

## Predict out of sample.
predictions_forest <- predict(multinomial_forest, X_test)
predictions_l1 <- predict(multinomial_l1, X_test)

## Compare predictions.
cbind(head(predictions_forest), head(predictions_l1))
```

ocf

Ordered Correlation Forest

Description

Nonparametric estimator for ordered non-numeric outcomes. The estimator modifies a standard random forest splitting criterion to build a collection of forests, each estimating the conditional probability of a single class.

Usage

```
ocf(
  y = NULL,
  X = NULL,
  honesty = FALSE,
  honesty.fraction = 0.5,
  inference = FALSE,
  alpha = 0,
  n.trees = 2000,
  mtry = ceiling(sqrt(ncol(X))),
  min.node.size = 5,
  max.depth = 0,
  replace = FALSE,
  sample.fraction = ifelse(replace, 1, 0.5),
  n.threads = 1
)
```

Arguments

y	Outcome vector.
X	Covariate matrix (no intercept).
honesty	Whether to grow honest forests.
honesty.fraction	Fraction of honest sample. Ignored if honesty = FALSE.
inference	Whether to extract weights and compute standard errors. The weights extraction considerably slows down the routine. honesty = TRUE is required for valid inference.
alpha	Controls the balance of each split. Each split leaves at least a fraction alpha of observations in the parent node on each side of the split.
n.trees	Number of trees.
mtry	Number of covariates to possibly split at in each node. Default is the square root of the number of covariates.
min.node.size	Minimal node size.
max.depth	Maximal tree depth. A value of 0 corresponds to unlimited depth, 1 to "stumps" (one split per tree).
replace	If TRUE, grow trees on bootstrap subsamples. Otherwise, trees are grown on random subsamples drawn without replacement.
sample.fraction	Fraction of observations to sample.
n.threads	Number of threads. Zero corresponds to the number of CPUs available.

Value

Object of class ocf.

Author(s)

Riccardo Di Francesco

References

- Di Francesco, R. (2023). Ordered Correlation Forest. arXiv preprint [arXiv:2309.08755](https://arxiv.org/abs/2309.08755).

See Also

[marginal_effects](#)

Examples

```
## Generate synthetic data.
set.seed(1986)

data <- generate_ordered_data(1000)
sample <- data$sample
Y <- sample$Y
X <- sample[, -1]

## Training-test split.
train_idx <- sample(seq_len(length(Y)), floor(length(Y) * 0.5))

Y_tr <- Y[train_idx]
X_tr <- X[train_idx, ]

Y_test <- Y[-train_idx]
X_test <- X[-train_idx, ]

## Fit ocf on training sample.
forests <- ocf(Y_tr, X_tr)

## We have compatibility with generic S3-methods.
print(forests)
summary(forests)
predictions <- predict(forests, X_test)
head(predictions$probabilities)
table(Y_test, predictions$classification)

## Compute standard errors. This requires honest forests.
honest_forests <- ocf(Y_tr, X_tr, honesty = TRUE, inference = TRUE)
head(honest_forests$predictions$standard.errors)

## Marginal effects.
me <- marginal_effects(forests, eval = "atmean")
print(me)
print(me, latex = TRUE)

## Compute standard errors. This requires honest forests.
honest_me <- marginal_effects(honest_forests, eval = "atmean", inference = TRUE)
honest_me$standard.errors
```

```
print(honest_me, latex = TRUE)
```

ordered_ml	<i>Ordered Machine Learning</i>
------------	---------------------------------

Description

Estimation strategy to estimate conditional choice probabilities for ordered non-numeric outcomes.

Usage

```
ordered_ml(y = NULL, X = NULL, learner = "forest", scale = TRUE)
```

Arguments

y	Outcome vector.
X	Covariate matrix (no intercept).
learner	String, either "forest" or "l1". Selects the base learner to estimate each expectation.
scale	Logical, whether to scale the covariates. Ignored if learner is not "l1".

Details

Ordered machine learning expresses conditional choice probabilities as the difference between the cumulative probabilities of two adjacent classes, which in turn can be expressed as conditional expectations of binary variables:

$$p_m(X_i) = \mathbb{E}[1(Y_i \leq m) | X_i] - \mathbb{E}[1(Y_i \leq m - 1) | X_i]$$

Then we can separately estimate each expectation using any regression algorithm and pick the difference between the m-th and the (m-1)-th estimated surfaces to estimate conditional probabilities.

`ordered_ml` combines this strategy with either regression forests or penalized logistic regression with an L1 penalty, according to the user-specified parameter `learner`.

If `learner == "forest"`, then the `orf` function is called from an external package, as this estimator has already been proposed by Lechner and Okasa (2019).

If `learner == "l1"`, the penalty parameters are chosen via 10-fold cross-validation and `model.matrix` is used to handle non-numeric covariates. Additionally, if `scale == TRUE`, the covariates are scaled to have zero mean and unit variance.

Value

Object of class `oml`.

Author(s)

Riccardo Di Francesco

References

- Di Francesco, R. (2023). Ordered Correlation Forest. arXiv preprint [arXiv:2309.08755](https://arxiv.org/abs/2309.08755).

See Also

[multinomial_ml](#), [ocf](#)

Examples

```
## Generate synthetic data.
set.seed(1986)

data <- generate_ordered_data(1000)
sample <- data$sample
Y <- sample$Y
X <- sample[, -1]

## Training-test split.
train_idx <- sample(seq_len(length(Y)), floor(length(Y) * 0.5))

Y_tr <- Y[train_idx]
X_tr <- X[train_idx, ]

Y_test <- Y[-train_idx]
X_test <- X[-train_idx, ]

## Fit ordered machine learning on training sample using two different learners.
ordered_forest <- ordered_ml(Y_tr, X_tr, learner = "forest")
ordered_l1 <- ordered_ml(Y_tr, X_tr, learner = "l1")

## Predict out of sample.
predictions_forest <- predict(ordered_forest, X_test)
predictions_l1 <- predict(ordered_l1, X_test)

## Compare predictions.
cbind(head(predictions_forest), head(predictions_l1))
```

predict.mml

Prediction Method for mml Objects

Description

Prediction method for class mml.

Usage

```
## S3 method for class 'mml'  
predict(object, data = NULL, ...)
```

Arguments

object	An mml object.
data	Data set of class <code>data.frame</code> . It must contain the same covariates used to train the base learners. If data is <code>NULL</code> , then <code>object\$X</code> is used.
...	Further arguments passed to or from other methods.

Details

If `object$learner == "l1"`, then `model.matrix` is used to handle non-numeric covariates. If we also have `object$scaling == TRUE`, then data is scaled to have zero mean and unit variance.

Value

Matrix of predictions.

Author(s)

Riccardo Di Francesco

References

- Di Francesco, R. (2023). Ordered Correlation Forest. arXiv preprint [arXiv:2309.08755](https://arxiv.org/abs/2309.08755).

See Also

[multinomial_ml](#), [ordered_ml](#)

Examples

```
## Generate synthetic data.  
set.seed(1986)  
  
data <- generate_ordered_data(1000)  
sample <- data$sample  
Y <- sample$Y  
X <- sample[, -1]  
  
## Training-test split.  
train_idx <- sample(seq_len(length(Y)), floor(length(Y) * 0.5))  
  
Y_tr <- Y[train_idx]  
X_tr <- X[train_idx, ]  
  
Y_test <- Y[-train_idx]  
X_test <- X[-train_idx, ]
```

```
## Fit multinomial machine learning on training sample using two different learners.
multinomial_forest <- multinomial_ml(Y_tr, X_tr, learner = "forest")
multinomial_l1 <- multinomial_ml(Y_tr, X_tr, learner = "l1")

## Predict out of sample.
predictions_forest <- predict(multinomial_forest, X_test)
predictions_l1 <- predict(multinomial_l1, X_test)

## Compare predictions.
cbind(head(predictions_forest), head(predictions_l1))
```

predict.ocf

Prediction Method for ocf Objects

Description

Prediction method for class [ocf](#).

Usage

```
## S3 method for class 'ocf'
predict(object, data = NULL, type = "response", ...)
```

Arguments

object	An ocf object.
data	Data set of class <code>data.frame</code> . It must contain at least the same covariates used to train the forests. If data is <code>NULL</code> , then <code>object\$full_data</code> is used.
type	Type of prediction. Either <code>"response"</code> or <code>"terminalNodes"</code> .
...	Further arguments passed to or from other methods.

Details

If `type == "response"`, the routine returns the predicted conditional class probabilities and the predicted class labels. If forests are honest, the predicted probabilities are honest.

If `type == "terminalNodes"`, the IDs of the terminal node in each tree for each observation in data are returned.

Value

Desired predictions.

Author(s)

Riccardo Di Francesco

References

- Di Francesco, R. (2023). Ordered Correlation Forest. arXiv preprint [arXiv:2309.08755](https://arxiv.org/abs/2309.08755).

See Also

[ocf](#), [marginal_effects](#)

Examples

```
## Generate synthetic data.
set.seed(1986)

data <- generate_ordered_data(1000)
sample <- data$sample
Y <- sample$Y
X <- sample[, -1]

## Training-test split.
train_idx <- sample(seq_len(length(Y)), floor(length(Y) * 0.5))

Y_tr <- Y[train_idx]
X_tr <- X[train_idx, ]

Y_test <- Y[-train_idx]
X_test <- X[-train_idx, ]

## Fit ocf on training sample.
forests <- ocf(Y_tr, X_tr)

## Predict on test sample.
predictions <- predict(forests, X_test)
head(predictions$probabilities)
predictions$classification

## Get terminal nodes.
predictions <- predict(forests, X_test, type = "terminalNodes")
predictions$forest.1[1:10, 1:20] # Rows are observations, columns are forests.
```

predict.oml

Prediction Method for oml Objects

Description

Prediction method for class oml.

Usage

```
## S3 method for class 'oml'  
predict(object, data = NULL, ...)
```

Arguments

object	An oml object.
data	Data set of class <code>data.frame</code> . It must contain the same covariates used to train the base learners. If data is NULL, then <code>object\$X</code> is used.
...	Further arguments passed to or from other methods.

Details

If `object$learner == "l1"`, then `model.matrix` is used to handle non-numeric covariates. If we also have `object$scaling == TRUE`, then data is scaled to have zero mean and unit variance.

Value

Matrix of predictions.

Author(s)

Riccardo Di Francesco

References

- Di Francesco, R. (2023). Ordered Correlation Forest. arXiv preprint [arXiv:2309.08755](https://arxiv.org/abs/2309.08755).

See Also

[multinomial_ml](#), [ordered_ml](#)

Examples

```
## Generate synthetic data.  
set.seed(1986)  
  
data <- generate_ordered_data(1000)  
sample <- data$sample  
Y <- sample$Y  
X <- sample[, -1]  
  
## Training-test split.  
train_idx <- sample(seq_len(length(Y)), floor(length(Y) * 0.5))  
  
Y_tr <- Y[train_idx]  
X_tr <- X[train_idx, ]  
  
Y_test <- Y[-train_idx]  
X_test <- X[-train_idx, ]
```

```
## Fit ordered machine learning on training sample using two different learners.
ordered_forest <- ordered_ml(Y_tr, X_tr, learner = "forest")
ordered_l1 <- ordered_ml(Y_tr, X_tr, learner = "l1")

## Predict out of sample.
predictions_forest <- predict(ordered_forest, X_test)
predictions_l1 <- predict(ordered_l1, X_test)

## Compare predictions.
cbind(head(predictions_forest), head(predictions_l1))
```

print.ocf

Print Method for ocf Objects

Description

Prints an `ocf` object.

Usage

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

Arguments

`x` An `ocf` object.
`...` Further arguments passed to or from other methods.

Value

Prints an `ocf` object.

Author(s)

Riccardo Di Francesco

References

- Di Francesco, R. (2023). Ordered Correlation Forest. arXiv preprint [arXiv:2309.08755](https://arxiv.org/abs/2309.08755).

See Also

`ocf`

Examples

```
## Generate synthetic data.
set.seed(1986)

data <- generate_ordered_data(1000)
sample <- data$sample
Y <- sample$Y
X <- sample[, -1]

## Fit ocf.
forests <- ocf(Y, X)

## Print.
print(forests)
```

print.ocf.marginal *Print Method for ocf.marginal Objects*

Description

Prints an ocf.marginal object.

Usage

```
## S3 method for class 'ocf.marginal'
print(x, latex = FALSE, ...)
```

Arguments

x	An ocf.marginal object.
latex	If TRUE, prints LATEX code.
...	Further arguments passed to or from other methods.

Details

Compilation of the LATEX code requires the following packages: booktabs, float, adjustbox. If standard errors have been estimated, they are printed in parenthesis below each point estimate.

Value

Prints an ocf.marginal object.

Author(s)

Riccardo Di Francesco

References

- Di Francesco, R. (2023). Ordered Correlation Forest. arXiv preprint [arXiv:2309.08755](https://arxiv.org/abs/2309.08755).

See Also

[ocf](#), [marginal_effects](#).

Examples

```
## Generate synthetic data.
set.seed(1986)

data <- generate_ordered_data(1000)
sample <- data$sample
Y <- sample$Y
X <- sample[, -1]

## Fit ocf.
forests <- ocf(Y, X)

## Marginal effects at the mean.
me <- marginal_effects(forests, eval = "atmean")
print(me)
print(me, latex = TRUE)

## Add standard errors.
honest_forests <- ocf(Y, X, n.trees = 4000, honesty = TRUE)
honest_me <- marginal_effects(honest_forests, eval = "atmean", inference = TRUE)
print(honest_me, latex = TRUE)
```

summary.ocf

Summary Method for ocf Objects

Description

Summarizes an [ocf](#) object.

Usage

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

Arguments

object	An ocf object.
...	Further arguments passed to or from other methods.

Value

Summarizes an `ocf` object.

Author(s)

Riccardo Di Francesco

References

- Di Francesco, R. (2023). Ordered Correlation Forest. arXiv preprint [arXiv:2309.08755](https://arxiv.org/abs/2309.08755).

See Also

`ocf`, `marginal_effects`

Examples

```
## Generate synthetic data.
set.seed(1986)

data <- generate_ordered_data(1000)
sample <- data$sample
Y <- sample$Y
X <- sample[, -1]

## Fit ocf.
forests <- ocf(Y, X)

## Summary.
summary(forests)
```

summary.ocf.marginal *Summary Method for ocf.marginal Objects*

Description

Summarizes an `ocf.marginal` object.

Usage

```
## S3 method for class 'ocf.marginal'
summary(object, latex = FALSE, ...)
```

Arguments

<code>object</code>	An <code>ocf.marginal</code> object.
<code>latex</code>	If TRUE, prints LATEX code.
<code>...</code>	Further arguments passed to or from other methods.

Details

Compilation of the LATEX code requires the following packages: booktabs, float, adjustbox. If standard errors have been estimated, they are printed in parenthesis below each point estimate.

Value

Summarizes an `ocf.marginal` object.

Author(s)

Riccardo Di Francesco

References

- Di Francesco, R. (2023). Ordered Correlation Forest. arXiv preprint [arXiv:2309.08755](https://arxiv.org/abs/2309.08755).

See Also

[ocf](#), [marginal_effects](#).

Examples

```
## Generate synthetic data.
set.seed(1986)

data <- generate_ordered_data(1000)
sample <- data$sample
Y <- sample$Y
X <- sample[, -1]

## Fit ocf.
forests <- ocf(Y, X)

## Marginal effects at the mean.
me <- marginal_effects(forests, eval = "atmean")
summary(me)
summary(me, latex = TRUE)

## Add standard errors.
honest_forests <- ocf(Y, X, honesty = TRUE)
honest_me <- marginal_effects(honest_forests, eval = "atmean", inference = TRUE)
summary(honest_me, latex = TRUE)
```

tree_info	<i>Tree Information in Readable Format</i>
-----------	--

Description

Extracts tree information from a `ocf` forest object.

Usage

```
tree_info(object, tree = 1)
```

Arguments

object	ocf forest object.
tree	Number of the tree of interest.

Details

Nodes and variables IDs are 0-indexed, i.e., node 0 is the root node.

All values smaller than or equal to `splitval` go to the left and all values larger go to the right.

Value

A data frame with the following columns:

nodeID	Node IDs.
leftChild	IDs of the left child node.
rightChild	IDs of the right child node.
splitvarID	IDs of the splitting variable.
splitvarName	Name of the splitting variable.
splitval	Splitting value.
terminal	Logical, TRUE for terminal nodes.
prediction	One column with the predicted conditional class probabilities.

Author(s)

Riccardo Di Francesco

References

- Di Francesco, R. (2023). Ordered Correlation Forest. arXiv preprint [arXiv:2309.08755](https://arxiv.org/abs/2309.08755).

See Also

[ocf](#)

Examples

```
## Generate synthetic data.
set.seed(1986)

data <- generate_ordered_data(1000)
sample <- data$sample
Y <- sample$Y
X <- sample[, -1]

## Fit ocf.
forests <- ocf(Y, X)

## Extract information from tenth tree of first forest.
info <- tree_info(forests$forests.info$forest.1, tree = 10)
head(info)
```


Index

classification_error, [7](#)
classification_error
 (mean_squared_error), [5](#)

generate_ordered_data, [2](#)

marginal_effects, [4](#), [4](#), [11](#), [16](#), [20–22](#)
mean_absolute_error, [6](#)
mean_absolute_error
 (mean_squared_error), [5](#)
mean_ranked_score, [6](#), [7](#)
mean_ranked_score (mean_squared_error),
 [5](#)
mean_squared_error, [5](#), [6](#)
model.matrix, [8](#), [12](#), [14](#), [17](#)
multinomial_ml, [8](#), [8](#), [13](#), [14](#), [17](#)

ocf, [3–5](#), [9](#), [9](#), [13](#), [15](#), [16](#), [18](#), [20–23](#)
ordered_ml, [9](#), [12](#), [12](#), [14](#), [17](#)
orf, [12](#)

predict.mml, [13](#)
predict.ocf, [15](#)
predict.oml, [16](#)
print.ocf, [18](#)
print.ocf.marginal, [19](#)

summary.ocf, [20](#)
summary.ocf.marginal, [21](#)

tree_info, [23](#)