Package: randomPlantedForest (via r-universe)

November 4, 2024

Title Random Planted Forest: A Directly Interpretable Tree Ensemble

Version 0.2.1.9000

Description An implementation of the Random Planted Forest algorithm for directly interpretable tree ensembles based on a functional ANOVA decomposition.

License Apache License (>= 2)

URL https://github.com/PlantedML/randomPlantedForest, http://plantedml.com/randomPlantedForest/

BugReports https://github.com/PlantedML/randomPlantedForest/issues

Imports checkmate, data.table, hardhat, methods, Rcpp, utils

Suggests knitr, mvtnorm, recipes, rmarkdown, testthat (>= 3.0.0)

LinkingTo Rcpp

VignetteBuilder knitr

Config/Needs/coverage covr

Config/Needs/website patchwork, ggplot2

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

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

RemoteUrl https://github.com/PlantedML/randomPlantedForest

RemoteRef HEAD

RemoteSha 41fe7eef99cfc60fbc04f6a08d30a932ffc097e0

Contents

`predict.rpf` *Random Planted Forest Predictions*

Description

Random Planted Forest Predictions

Usage

```
## S3 method for class 'rpf'
predict(
  object,
  new_data,
  type = ifelse(object$mode == "regression", "numeric", "prob"),
  ...
)
```

Arguments

`object`	A fit object of class `rpf`.
`new_data`	Data for new observations to predict.
`type`	`"numeric"` for regression outcomes, `"class"` for class predictions or `"prob"` for probability predictions. For classification and `loss = "L1"` or `"L2"`, `"numeric"` yields raw predictions which are not guaranteed to be valid probabilities in `[0, 1]`. For `type = "prob"`, these are truncated to ensure this property. If `loss` is `"logit"` or `"exponential"`, `type = "link"` is an alias for `type = "numeric"`, as in this case the raw predictions have the additional interpretation similar to the linear predictor in a `glm`.
`...`	Unused.

Value

For regression: A `tbl` with column `.pred` with the same number of rows as `new_data`.

For classification: A `tbl` with one column for each level in `y` containing class probabilities if `type = "prob"`. For `type = "class"`, one column `.pred` with class predictions is returned. For `type = "numeric"` or `"link"`, one column `.pred` with raw predictions.

Examples

```
# Regression with L2 loss
rpfit <- rpf(y = mtcars$mpg, x = mtcars[, c("cyl", "wt")])
predict(rpfit, mtcars[, c("cyl", "wt")])
```

`predict_components` *Extract predicted components from a Random Planted Forest*

Description

Prediction components are a functional decomposition of the model prediction. The sum of all components equals the overall predicted value for an observation.

Usage

```
predict_components(object, new_data, max_interaction = NULL, predictors = NULL)
```

Arguments

`object`	A fit object of class `rpf`.
`new_data`	Data for new observations to predict.
`max_interaction`	`integer` or `NULL`: Maximum degree of interactions to consider. Default will use the `max_interaction` parameter from the `rpf` object. Must be between 1 (main effects only) and the `max_interaction` of the `rpf` object.
`predictors`	`character` or `NULL`: Vector of one or more column names of predictor variables in `new_data` to extract components for. If `NULL`, all variables and their interactions are returned.

Details

Extracts all possible components up to `max_interaction` degrees, up to the value set when calling `rpf()`. The intercept is always included. Optionally `predictors` can be specified to only include components including the given variables. If `max_interaction` is greater than `length(predictors)`, the `max_interaction` will be lowered accordingly.

Value

A `list` with elements:

- `m` (`data.table`): Components for each main effect and interaction term, representing the functional decomposition of the prediction. All components together with the intercept sum up to the prediction. For multiclass classification, the number of output columns is multiplied by the number of levels in the outcome.
- `intercept` (`numeric(1)`): Expected value of the prediction.
- `x` (`data.table`): Copy of `new_data` containing predictors selected by `predictors`.
- `target_levels` (`character`): For multiclass classification only: Vector of target levels which can be used to disassemble `m`, as names include both term and target level.

Note

Depending on the number of predictors and `max_interaction`, the number of components will increase drastically to `sum(choose(ncol(new_data), seq_len(max_interaction)))`.

Examples

```
# Regression task, only some predictors
train <-  mtcars[1:20, 1:4]
test <-  mtcars[21:32, 1:4]

set.seed(23)
rpfit <- rpf(mpg ~ ., data = train, max_interaction = 3, ntrees = 30)

# Extract all components, including main effects and interaction terms up to `max_interaction`
(components <- predict_components(rpfit, test))

# sums to prediction
cbind(
  m_sum = rowSums(components$m) + components$intercept,
  prediction = predict(rpfit, test)
)

# Only get components with interactions of a lower degree, ignoring 3-way interactions
predict_components(rpfit, test, max_interaction = 2)

# Only retrieve main effects
(main_effects <- predict_components(rpfit, test, max_interaction = 1))

# The difference is the combined contribution of interaction effects
cbind(
  m_sum = rowSums(main_effects$m) + main_effects$intercept,
  prediction = predict(rpfit, test)
)
```

`print.rpf` *Print an rpf fit*

Description

Print an rpf fit

Usage

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

Arguments

x	And object of class rpf.
...	Further arguments passed to or from other methods.

Value

Invisibly: x.

See Also

rpf.

Examples

```
rpf(mpg ~ cyl + wt + drat, data = mtcars, max_interaction = 2, ntrees = 10)
```

---

print.rpf_forest *Compact printing of forest structures*

Description

These methods are provided to avoid flooding the console with long nested lists containing tree structures. Note

Usage

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

## S3 method for class 'rpf_forest'
str(object, ...)
```

Arguments

x	Object of class rpf_forest
...	Further arguments passed to or from other methods.
object	Object of class rpf_forest

See Also

rpf

Examples

```
rpfit <- rpf(mpg ~ cyl + wt, data = mtcars, ntrees = 10)
print(rpfit$forest)
str(rpfit$forest)
```

purify *Purify a Random Planted Forest*

Description

TODO: Explain what this does

Usage

```
purify(x, ...)

## Default S3 method:
purify(x, ...)

## S3 method for class 'rpf'
purify(x, ...)

is_purified(x)
```

Arguments

x	And object of class rpf.
...	(Unused)

Details

Unless rpf() is called with purify = TRUE, the forest has to be purified after fit to ensure the components extracted by predict_components() are valid. predict_components() will automatically purify a forest if is_purified() reports FALSE.

Value

Invisibly: The rpf object.

Examples

```
rpfit <- rpf(mpg ~., data = mtcars, max_interaction = 2, ntrees = 10)
purify(rpfit)
```

rpf *Random Planted Forest*

Description

Random Planted Forest

Usage

```
rpf(x, ...)

## S3 method for class 'data.frame'
rpf(
  x,
  y,
  max_interaction = 1,
  ntrees = 50,
  splits = 30,
  split_try = 10,
  t_try = 0.4,
  deterministic = FALSE,
  nthreads = 1,
  purify = FALSE,
  cv = FALSE,
  loss = "L2",
  delta = 0,
  epsilon = 0.1,
  ...
)

## S3 method for class 'matrix'
rpf(
  x,
  y,
  max_interaction = 1,
  ntrees = 50,
  splits = 30,
  split_try = 10,
  t_try = 0.4,
  deterministic = FALSE,
  nthreads = 1,
  purify = FALSE,
  cv = FALSE,
  loss = "L2",
  delta = 0,
  epsilon = 0.1,
  ...
```

```
)

## S3 method for class 'formula'
rpf(
  formula,
  data,
  max_interaction = 1,
  ntrees = 50,
  splits = 30,
  split_try = 10,
  t_try = 0.4,
  deterministic = FALSE,
  nthreads = 1,
  purify = FALSE,
  cv = FALSE,
  loss = "L2",
  delta = 0,
  epsilon = 0.1,
  ...
)

## S3 method for class 'recipe'
rpf(
  x,
  data,
  max_interaction = 1,
  ntrees = 50,
  splits = 30,
  split_try = 10,
  t_try = 0.4,
  deterministic = FALSE,
  nthreads = 1,
  purify = FALSE,
  cv = FALSE,
  loss = "L2",
  delta = 0,
  epsilon = 0.1,
  ...
)
```

Arguments

x, data	Feature matrix, or data.frame, or recipe.
...	(Unused).
y	Target vector for use with x. The class of y (either numeric or factor) determines if regression or classification will be performed.
max_interaction	[1]: Maximum level of interaction determining maximum number of split di-

	mensions for a tree. The default 1 corresponds to main effects only. If 0, the number fo columns in x is used, i.e. for 10 predictors, this is equivalent to setting max_interaction = 10.
ntrees	[50]: Number of trees generated per family.
splits	[30]: Number of splits performed for each tree family.
split_try	[10]: Number of split points to be considered when choosing a split candidate.
t_try	[0.4]: A value in (0,1] specifying the proportion of viable split-candidates in each round.
deterministic	[FALSE]: Choose whether approach deterministic or random.
nthreads	[1L]: Number of threads used for computation, defaulting to serial execution.
purify	[FALSE]: Whether the forest should be purified. Set to TRUE to enable components extract with predict_components() are valid. Can be achieved after fitting with purify().
cv	[FALSE]: Determines if cross validation is performed.
loss	["L2"]: For regression, only "L2" is supported. For classification, "L1", "logit" and "exponential" are also available. "exponential" yields similar results as "logit" while being significantly faster.
delta	[0]: Only used if loss is "logit" or "exponential". Proportion of class membership is truncated to be smaller 1-delta when calculating the loss to determine the optimal split.
epsilon	[0.1]: Only used if loss = "logit" or "exponential". Proportion of class membership is truncated to be smaller 1-epsilon when calculating the fit in a leaf.
formula	Formula specification, e.g. y ~ x1 + x2.

Value

Object of class "rpf" with model object contained in $fit.

Examples

```
# Regression with x and y
rpfit <- rpf(x = mtcars[, c("cyl", "wt")], y = mtcars$mpg)

# Regression with formula
rpfit <- rpf(mpg ~ cyl + wt, data = mtcars)
```

Index