Package 'fru'

November 16, 2026

Title A Blazing Fast Implementation of Random Forest

Version 1.0.1

Description Yet another implementation of the Random Forest method by Breiman (2001) <doi:10.1023/A:1010933404324>, written in Rust and tailored towards stability, correctness, efficiency and scalability on modern multi-core machines. Handles both classification and regression, as well as provides permutation feature importance via a novel, highly optimised algorithm.

URL https://gitlab.com/mbq/fru

SystemRequirements Cargo (Rust's package manager) >= 1.85, rustc >= 1.85

License GPL-3

Encoding UTF-8

Config/roxygen2/version 8.1.0

NeedsCompilation yes

Author Miron Bartosz Kursa [aut, cre] (ORCID: <https://orcid.org/0000-0001-7672-648X>), Krzysztof Piotr Piwoński [aut] (ORCID: <https://orcid.org/0000-0002-0977-5959>)

Maintainer Miron Bartosz Kursa <m@mbq.me>

Repository CRAN

Date/Publication 2026-11-16 03:10:02 UTC

Contents

extract_forest 2
fru 2
importance 4
predict.fru 5
print.fru 7
solidify 7

Index **9**

extract_forest *Extract the forest*

Description

Extracts the whole decision forest as a left-first, depth-first walk over all vertices.

Usage

```
extract_forest(x)
```

Arguments

x	A model to convert; has to hold the forest (`forest=TRUE` flag passed to `fru`).

Value

A data frame with the forest structure. Each row represents a step in a left-first, depth-first walk over the forest. The `Feature` column holds, for branches, the feature used for a split, or NA, for leaves. Similarly, `Threshold` and `Subset` columns hold the splitting criterion for branches; they exists only when holding any data. For a numerical or integer split, observations with values strictly larger than threshold are sent left. For a subset split, observations with values in the threshold subsets are sent left. Logical splits have a fixed criterion, `TRUE`s are sent left. This way, they have no corresponding criterion column. Finally, leaf visits have their vote stored in the `Vote colum`.

Note

This function will solidify the model object.

fru *Train the fru model*

Description

Fru is an implementation of Leo Breiman's Random Forest (tm) method. It fits an ensemble of decision trees built on bootstrap resamples of observations and additionally permuted by constraining split optimisation to a random subset of features. The ensemble prediction is than established from individual trees by voting. Thanks to its construction, the model can also provide a cross-validation-like internal approximation of error, so called out-of-bag predictions, as well as importance scores for features.

Usage

```
fru(
  x,
  y,
  trees = 500L,
  tries,
  forest = FALSE,
  oob = TRUE,
  importance = FALSE,
  solidify = FALSE,
  threads = 0L
)
```

Arguments

x	Data frame containing predictors; must only contain logical, numeric, integer or factor columns, without NAs.
y	Decision; either a factor or logical, for classification, or numeric, for regression. Integer decision will be silently converted into a real vector and treated as such afterwards. NA values are not accepted.
trees	Number of trees to grow, a single number larger than zero. Also called `ntree` in other software. 500 by default, in principle should be set to be big enough to stabilise the outputs of the forest, either prediction accuracy or importance; generally, bigger sets will need more trees, and it is unlikely that overshooting ensemble size will hurt the model in a statistically significant way.
tries	Number of features to try at each split, a single number larger than zero and not larger than the number of columns in `x`. Also called `mtry` in other software. By default, set to the rounded square root of the number of features. It is unlikely this needs tweaking; increasing this value leads to a more accurate decision trees, but in turn makes them more correlated, spoiling the ensemble effect.
forest	If set to `TRUE`, the forest object is returned and can be used for prediction.
oob	If set to `TRUE`, out-of-bag (OOB) predictions will be calculated.
importance	If set to `TRUE`, importance scores will be calculated.
solidify	If set to `TRUE`, the forest object will use more memory but will survive serialisation, in particular when saved by `save`, `saveRDS` or when sent between processes. This can be done later with `solidify`, unless the model structure was already lost.
threads	Number of threads to use; by default, or when set to 0, fru will try to use all available computing cores.

Details

In comparison to similar packages, fru is a tailored towards stability, correctness, efficiency and scalability on modern multithreaded machines, providing solid foundation for large data analysis, higher-level methods or production pipeline. To this end, fru exposes only the original hyperparameters and provides only the permutational importance, though calculated with a novel algorithm that alleviates its greater computational burden.

Fru accepts logical, numeric (including integer) and factor features; NAs are not allowed and will result in error. Logical features are always split into false/true groups without optimisation, yet are scored via weighted Gini impurity (for classification) or variance reduction (for regression), in order to be compared with splits on other features. For numerical features, threshold value is optimised by an exhaustive scan of the criterion above; real values get threshold as a mid-point between values around the split, while integer values as a minimal of the two. In case of a tie in the score, a smaller threshold is used. Ordered factors or factors with six or more levels are treated as numerical, so follow the above procedure. Unordered factors with five or less levels are split by finding a level partition into two subsets via an exhaustive scan of all possibilities, scored, as above, by Gini impurity or variance drop, depending on the forest type.

The maximal tree depth is hard-coded to 512; a critical sample size that triggers branch termination into leaf is one for classification and four for regression; this means that regression needs at least ten objects to be practical. Leaves may be formed from larger samples in same cases, for instance when no split can be found based on the feature samples; in this case, for classification, random tie breaking is used.

Fru uses its own PRNG, the pcg32 method by Melissa E. O'Neill, for its capacity to produce reasonably decorrelated streams, which are used to provide reproducibility of the output in parallel scenarios, regardless of the number of threads. Namely, fru guarantees that the same trees will be fit for the same input and random seed, although their order may differ. Thus, OOB predictions and importance scores will be the same up to numerical errors. PRNG is used in training and prediction on new data; generator is seeded from the R generator, thus standard R interface of `set.seed` should be used to control it.

Value

The fitted model, an object of a class `fru`.

References

Kursa Miron B., Piwoński Krzysztof P. (2026). *fru: Fast random forest implementation*, SoftwareX 35, 102918.

Breiman Leo (2001). *Random Forests*, Machine Learning 45, 5-32.

O'Neil Melissa E. (2014). *PCG: A Family of Simple Fast Space-Efficient Statistically Good Algorithms for Random Number Generation*, HMC-CS-2014-0905.

Examples

```
set.seed(1)
data(iris)
fru(iris[,-5],iris[,5],threads=2)
```

`importance` *Extract importance*

Description

Extracts importance from the `fru` model.

Usage

```
importance(x, ...)

## S3 method for class 'fru'
importance(x, scale = FALSE, ...)
```

Arguments

x	A model from which importance scores should be extracted; has to hold importance scores (importance=TRUE flag passed to fru).
...	Ignored.
scale	If TRUE, importance scores will be scaled their standard deviation over the ensemble.

Value

A vector of importance scores, in the order and named as columns were in the training data.

Note

Other packages often scale importance by its standard error estimate, thus producing scales importance values square root of tree count times larger than fru. If you get a "non applicable method" error, this method was probably shadowed by other package. Use fru:::importance to call this function explicitly.

Examples

```
set.seed(1)
data(iris)
fru(iris[,-5],iris[,5],threads=2,importance=TRUE)->model
importance(model)
```

---

predict.fru *Predict with the fru model*

Description

Either predicts a given new data or returns the OOB predictions of the model; optionally, for classification forests, returns raw votes for each decision class.

Usage

```
## S3 method for class 'fru'
predict(object, x, votes = FALSE, threads = 0L, ...)
```

Arguments

`object`	A model used for prediction; has to hold the forest (`forest=TRUE` flag passed to `fru`) to make predictions on new data, or has to have OOB scores (`oob=TRUE` flag passed to `fru`) to return OOB scores.
`x`	Data frame to predict; if missing or NULL, the method will return OOB scores.
`votes`	If set to `TRUE`, changes the output to sums of votes cast by the ensemble on each class; useful as a prediction confidence score, for instance for ROC analysis. Only makes sense for classification; passing this flag together with regression forest will throw an error.
`threads`	Number of threads to use; by default, or when set to 0, fru will try to use all available computing cores.
`...`	Ignored.

Details

If given, new data has to hold the same features as the training data, and the method will match them by name (order is irrelevant, additional features will be ignored); matched features have to be of the same type. Moreover, factor features have to have exactly the same levels in the same order as in training; this will be checked.

The voting in classification case may lead to ties, in which case predict will use PRNG to resolve them. In the OOB mode, the constant seed is used, so that OOB scores for the same forest model will always be the same, mimicking the behaviour of other packages which usually calculate predictions during training and store them with ties resolved at that time. For new data prediction, PRNG is seeded from R's random state, so, in principle, ties will be resolved differently on each prediction. If determinism is desired, it is best to use the votes output in which ties are evident. Regression is performed using leaf averages, which is a deterministic process (not counting numerical issues possibly caused by nondeterministic order in which trees are produced when using multi-threading).

The OOB predictions may contain NAs when a given object was not an OOB object of any tree, which may happen for small ensembles (in particular surely when `trees=1`). Similarly, the sums of OOB votes for each object will not sum up to the ensemble size, but will for new data prediction.

By the nature of the method, new data prediction for the training data is usually close to perfect reproduction of the training decision; it is basically useless for any practical use.

This method checks matches the input structure with the training data structure retained in the object, which may take some time, especially when data is large or short prediction latency is required. In that case, one may use the non-exported `unsafe_fru_predict` function which expects `x` to be exactly in the same form as training, jumps straight to the compiled code and returns the predictions in the raw form (classes are level indices, vote matrix is unrolled, etc.).

Value

For a default of `votes=FALSE`, a vector with a prediction for either each row of `x`, or, when not given, an OOB approximated prediction for each row of the original training data. For `votes=TRUE`, a data frame with as many columns as decision classes, rows corresponding to rows of `x` or training data, and cells with the counts of votes per each class.

Examples

```
set.seed(1)
data(iris)
iris[c(TRUE,FALSE),]->iris_train
iris[c(FALSE,TRUE),]->iris_test
fru(iris_train[,-5],iris_train[,5],threads=2,forest=TRUE)->model
print(model)
table(predict(model,iris_test,threads=2),iris_test$Species)
```

---

print.fru *Print the fru object*

Description

Prints the basic information about the fitted model, in particular the OOB error estimated (if enabled previously).

Usage

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

Arguments

x	Model to print.
...	Ignored.

Value

Invisibly, the same object x.

---

solidify *Solidify a given fru object*

Description

Forces a model to be solidified, so that it would survive through saving into RDS or sending over a network, etc. The downside is that the forest (and/or OOB scores or importance) will exist twice in the memory, and this process takes some time. The function converts the object in-place, thanks to the semantics of external pointers. No-op when given an object that is already serialised, either by solidify=TRUE flag passed to fru, due to a previous call to solidify or when deserialised.

Usage

```
solidify(x)
```

Arguments

x	The fru model object.

Value

Invisibly, the same object as x; the function is called for the side effect of modifying the object.

Index

extract_forest, 2

fru, 2

importance, 4

predict.fru, 5
print.fru, 7

solidify, 7