Package 'StratifiedRF'

October 12, 2022

Type Package

Title Builds Trees by Sampling Variables in Groups

Version 0.2.2

Author David Cortes <david.cortes.rivera@gmail.com>

Maintainer David Cortes <david.cortes.rivera@gmail.com>

Description Random Forest-like tree ensemble that works with groups of predictor variables. When building a tree, a number of variables is taken randomly from each group separately, thus ensuring that it considers variables from each group for the splits. Useful when rows contain information about different things (e.g. user information and product information) and it's not sensible to make a prediction with information from only one group of variables, or when there are far more variables from one group than the other and it's desired to have groups appear evenly on trees. Trees are grown using the C5.0 algorithm rather than the usual CART algorithm. Supports parallelization (multithreaded), missing values in predictors, and categorical variables (without doing One-Hot encoding in the processing). Can also be used to create a regular (non-stratified) Random Forest-like model, but made up of C5.0 trees and with some additional control options. As it's built with C5.0 trees, it works only for classification (not for regression).

Imports C50, dplyr, parallel, stats

License GPL-3

LazyData TRUE

RoxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2017-06-30 17:19:38 UTC

R topics documented:

`predict.stratified_rf` *Make predictions on new data*

Description

Make predictions from a stratified_rf model on new data.

Usage

```
## S3 method for class 'stratified_rf'
predict(object, data, type = "class",
  agg_type = "prob", vote_type = "simple", na.action = na.pass,
  threshold = NULL, ...)
```

Arguments

`object` A stratified_rf model.

`data` New data on which to make predictions (data.frame only). Must have the same names as the data used to build the model.

`type` Prediction type. Either "class" to get the predicted class or "prob" to get the voting scores for each class.

`agg_type` How to combine the predictions from individual trees. Either "prob" to average the probabilities output from each tree or "class" to count the final predictions from each.

`vote_type` How to weight the outputs from each tree. Either "simple" to average them, or "weighted" for a weighted average according to their OOB classification accuracy.

`na.action` Function indicating how to handle missing values (see the 'C50' documentation for details).

`threshold` Count only votes from trees whose out-of-bag classification accuracy is above this threshold. Must be a number between 0 and 1.

`...` other options (not currently used)

Details

Note that by default, for classification models the predictions are made quite differently from the original Random Forest algorithm.

See Also

'C50' library: `https://cran.r-project.org/package=C50`

Examples

```
data(iris)
groups <- list(c("Sepal.Length","Sepal.Width"),c("Petal.Length","Petal.Width"))
mtry <- c(1,1)
m <- stratified_rf(iris,"Species",groups,mtry,ntrees=2,multicore=FALSE)
predict(m,iris)
```

print.stratified_rf *Print summary statistics from a model*

Description

Print summary statistics from a model

Usage

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

Arguments

x	A stratified_rf model.
...	other options (not currently used)

Examples

```
data(iris)
groups <- list(c("Sepal.Length","Sepal.Width"),c("Petal.Length","Petal.Width"))
mtry <- c(1,1)
m <- stratified_rf(iris,"Species",groups,mtry,ntrees=2,multicore=FALSE)
print(m)
```

stratified_rf *Stratified Random Forest*

Description

Random Forest that works with groups of predictor variables. When building a tree, a number of variables is taken from each group separately. Useful when rows contain information about different things (e.g. user information and product information) and it's not sensible to make a prediction with information from only one group of variables, or when there are far more variables from one group than the other and it's desired to have groups appear evenly on trees.

Usage

```
stratified_rf(df, targetvar, groups, mtry = "auto", ntrees = 500,
  multicore = TRUE, class_quotas = NULL, sample_weights = NULL,
  fulldepth = TRUE, replacement = TRUE, c50_control = NULL,
  na.action = na.pass, drop_threshold = NULL)
```

Arguments

`df` Data to build the model (data.frame only).

`targetvar` String indicating the name of the target or outcome variable in the data. Character types will be coerced to factors.

`groups` Unnamed list, containing at each entry a group of variables (as a string vector with their names).

`mtry` A numeric vector indicating how many variables to take from each group when building each tree. If set to "auto" then, for each group, mtry=round(sqrt(m_total)*len(m_group)/len(m_to (with a minimum of 1 for each group).

`ntrees` Number of trees to grow. When setting multicore=TRUE, the number of trees should be a multiple of the number of cores, otherwise it will get rounded downwards to the nearest multiple.

`multicore` Whether to use multiple CPU cores to parallelize the construction of trees. Parallelization is done with the 'parallel' library's default settings.

`class_quotas` How many rows from each class to use in each tree (useful when there is a class imbalance). Must be a numeric vector or a named list with the number of desired rows to sample for each level of the target variable. Ignored when sample_weights is passed. Note that using more rows than the data originally had might result in incorrect out-of-bag error estimates.

`sample_weights` Probability of sampling each row when building a tree. Must be a numeric vector. If not defined, then all rows have the same probability. Note that, depending on the structure of the data, setting this might result in incorret out-of-bag error estimates.

`fulldepth` Whether to grow the trees to full depth. Ignored when passing c50_control.

`replacement` Whether to sample rows with replacement.

`c50_control` Custom parameters for growing trees. Must be a C5.0Control object compatible with the 'C50' package.

`na.action` A function indicating how to handle NAs. Default is to include missing values when building a tree (see 'C50' documentation).

`drop_threshold` Drop a tree whenever its resulting out-of-bag classification accuracy falls below a certain threshold specified here. Must be a number between 0 and 1.

Details

Note that while this algorithm forces each tree to consider possible splits with variables from all groups, it doesn't guarantee that they will end up having splits with variables from different groups.

The original Random Forest algorithm recommends a total number of sqrt(n_features), but this might not work so well when there are unequal groups of variables.

Implementation of everything outside the tree-building is in native R code, thus might be slow. Trees are grown using the C5.0 algorithm from the 'C50' library, thus it can be used for classification only (not for regression). Refer to the 'C50' library for any documentation about the tree-building algorithm.

See Also

'C50' library: https://cran.r-project.org/package=C50

Examples

```
data(iris)
groups <- list(c("Sepal.Length","Sepal.Width"),c("Petal.Length","Petal.Width"))
mtry <- c(1,1)
m <- stratified_rf(iris,"Species",groups,mtry,ntrees=2,multicore=FALSE)
summary(m)
```

---

summary.stratified_rf *Summary statistics from a model*

Description

Calculates error statistics for out-of-bag samples from a stratified_rf model.

Usage

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

Arguments

`object`	A stratified_rf model.
`...`	other options (not currently used)

Details

Predictions for a class are made by averaging class probabilities across trees rather than by a majority vote. All trees are weighted equally.

Examples

```
data(iris)
groups <- list(c("Sepal.Length","Sepal.Width"),c("Petal.Length","Petal.Width"))
mtry <- c(1,1)
m <- stratified_rf(iris,"Species",groups,mtry,ntrees=2,multicore=FALSE)
summary(m)
```

varimp_stratified_rf *Heuristic on variable importance*

Description

Heuristic on variable importance, taken as averages from the variable importances calculated for each tree.

Usage

```
varimp_stratified_rf(model, metric = "usage", agg_type = "simple")
```

Arguments

model	A stratified_rf model.
metric	How to calculate the variable importance from each tree. Either "usage" or "splits".
agg_type	How to aggregate the variable importances obtained from each tree. Either "simple" for a simple average, or "weighted" for an average weighted by each tree's accuracy.

Details

Methods are taken directly from the C5.0 trees. Currently doesn't support permutation tests.

Value

A named data frame with the importance score of each variable, sorted from largest to smallest.

Examples

```
data(iris)
groups <- list(c("Sepal.Length","Sepal.Width"),c("Petal.Length","Petal.Width"))
mtry <- c(1,1)
m <- stratified_rf(iris,"Species",groups,mtry,ntrees=2,multicore=FALSE)
varimp_stratified_rf(m)
```

Index