

Package ‘sirus’

September 2, 2019

Type Package

Title Stable and Interpretable RULe Set

Version 0.1.2

Date 2019-09-01

Author Clement Benard [aut, cre], Marvin N. Wright [ctb, cph]

Maintainer Clement Benard <clement.benard@safrangroup.com>

Description A classification algorithm based on random forests, which takes the form of a short list of rules. SIRUS combines the simplicity of decision trees with the predictivity of random forests for problems with low order interactions. The core aggregation principle of random forests is kept, but instead of aggregating predictions, SIRUS selects the most frequent nodes of the forest to form a stable rule ensemble model. The algorithm is fully described in the following article: Benard C., Biau G., da Veiga S., Scornet E. (2019) <arXiv:1908.06852>. This R package is a fork from the project ranger (<<https://github.com/imbs-hl/ranger>>).

License GPL-3

Imports Rcpp (>= 0.11.2), Matrix, ROCR, ggplot2

LinkingTo Rcpp

Depends R (>= 3.1)

Suggests survival, testthat

RoxygenNote 6.1.1

URL <https://gitlab.com/safrandrati/sirus>

BugReports <https://gitlab.com/safrandrati/sirus/issues>

NeedsCompilation yes

Repository CRAN

Date/Publication 2019-09-02 14:40:02 UTC

R topics documented:

sirus.cv	2
sirus.fit	3
sirus.plot.cv	5
sirus.predict	6
sirus.print	7

Index	8
--------------	----------

sirus.cv	<i>Estimation of p_0.</i>
----------	--

Description

Estimation by cross-validation of the hyperparameter p_0 used to select rules in `sirus.fit`. For a robust estimation, it is recommended to run multiple cross-validations, typically 30.

Usage

```
sirus.cv(data, y, nfold = 10, ncv = 30, num.rule.max = 25, q = 10,
  num.trees.step = 1000, alpha = 0.05, mtry = NULL,
  num.trees = NULL, num.threads = NULL, replace = TRUE,
  sample.fraction = NULL, verbose = TRUE, seed = NULL)
```

Arguments

<code>data</code>	Input dataframe, each row is an observation vector.
<code>y</code>	Binary response variable taking 0 and 1 values.
<code>nfold</code>	Number of folds in the cross-validation. Default is 10.
<code>ncv</code>	Number of repetitions of the cross-validation. Default is 30.
<code>num.rule.max</code>	Maximum number of rules of SIRUS model in the cross-validation grid. Default is 25.
<code>q</code>	Number of quantiles used for node splitting in the forest construction. Default is 10.
<code>num.trees.step</code>	Number of trees grown between two evaluations of the stopping criterion. Ignored if <code>num.trees</code> is provided.
<code>alpha</code>	Parameter of the stopping criterion for the number of trees: stability has to reach $1 - \alpha$ to stop the growing of the forest. Ignored if <code>num.trees</code> is provided.
<code>mtry</code>	Number of variables to possibly split at each node. Default is the number of variables divided by 3.
<code>num.trees</code>	Number of trees grown in the forest. If NULL (recommended), the number of trees is automatically set using a stability stopping criterion.
<code>num.threads</code>	Number of threads used to grow the forest. Default is number of CPUs available.
<code>replace</code>	Boolean. If true (default), sample with replacement.

sample.fraction	Fraction of observations to sample. Default is 1 for sampling with replacement and 0.632 for sampling without replacement.
verbose	Boolean. If true, information messages are printed.
seed	Random seed. Default is NULL, which generates the seed from R. Set to 0 to ignore the R seed.

Value

	Optimal value of p_0 with the elements
p_0	Optimal p_0 value.
error	1 - AUC of the SIRUS model with the optimal p_0 .
num.rules	Number of rules of the SIRUS model with the optimal p_0 .
error.grid.p0	Table with the full cross-validation results for a fine grid of p_0 : number of rules, stability, error (1-AUC).

Examples

```
## load sirus
require(sirus)

## prepare data
data <- iris
y <- rep(0, nrow(data))
y[data$Species == 'setosa'] = 1
data$Species <- NULL

## run cv
cv.grid <- sirus.cv(data, y, nfold = 3, ncv = 2, num.trees = 100)
```

sirus.fit

Fit SIRUS.

Description

Fit a SIRUS classification model for a given number of rules (10 by default) or a given p_0 . The number of trees is tuned automatically with a stopping criterion based on stability. The hyperparameter p_0 can be tuned using `sirus.cv`.

Usage

```
sirus.fit(data, y, num.rule = 10, p0 = NULL, num.rule.max = 25,
  q = 10, num.trees.step = 1000, alpha = 0.05, mtry = NULL,
  num.trees = NULL, num.threads = NULL, replace = TRUE,
  sample.fraction = ifelse(replace, 1, 0.632), verbose = TRUE,
  seed = NULL)
```

Arguments

<code>data</code>	Input dataframe, each row is an observation vector.
<code>y</code>	Binary response variable taking 0 and 1 values.
<code>num.rule</code>	Number of rules in SIRUS model. Default is 10. Ignored if a <code>p0</code> value is provided.
<code>p0</code>	Selection threshold on the frequency of appearance of a path in the forest. Default is NULL and <code>num.rule</code> is used to select rules.
<code>num.rule.max</code>	Maximum number of rules in SIRUS model. Ignored if <code>num.rule</code> is provided.
<code>q</code>	Number of quantiles used for node splitting in the forest construction.
<code>num.trees.step</code>	Number of trees grown between two evaluations of the stopping criterion. Ignored if <code>num.trees</code> is provided.
<code>alpha</code>	Parameter of the stopping criterion for the number of trees: stability has to reach $1 - \alpha$ to stop the growing of the forest. Ignored if <code>num.trees</code> is provided.
<code>mtry</code>	Number of variables to possibly split at each node. Default is the number of variables divided by 3.
<code>num.trees</code>	Number of trees grown in the forest. Default is NULL. If NULL (recommended), the number of trees is automatically set using a stability based stopping criterion.
<code>num.threads</code>	Number of threads used to grow the forest. Default is number of CPUs available.
<code>replace</code>	Boolean. If true (default), sample with replacement.
<code>sample.fraction</code>	Fraction of observations to sample. Default is 1 for sampling with replacement and 0.632 for sampling without replacement.
<code>verbose</code>	Boolean. If true, information messages are printed.
<code>seed</code>	Random seed. Default is NULL, which generates the seed from R. Set to 0 to ignore the R seed.

Value

SIRUS model with elements

<code>rules</code>	List of rules in SIRUS model.
<code>rules.out</code>	List of rule outputs. <code>rule.out</code> : the output mean whether the rule is satisfied or not. <code>supp.size</code> : the number of points inside and outside the rule.
<code>proba</code>	Frequency of occurrence of paths in the forest.
<code>paths</code>	List of selected paths.
<code>mean</code>	Mean output over the full training data. Default model output if no rule is selected.

Examples

```
## load sirius
require(sirius)

## prepare data
data <- iris
y <- rep(0, nrow(data))
y[data$Species == 'setosa'] = 1
data$Species <- NULL

## fit sirius
sirius.m <- sirius.fit(data, y)
```

`sirius.plot.cv`*Plot SIRUS cross-validation.*

Description

Plot SIRUS cross-validation path: 1-AUC and stability versus the number of rules when p_0 varies.

Usage

```
sirius.plot.cv(sirius.cv.grid, num.rule.max = 25)
```

Arguments

`sirius.cv.grid` Cross-validation results returned by `sirius.cv`.
`num.rule.max` Upper limit on the number of rules for the x-axis. Default is 25.

Value

Plots of cross-validation results.

`error` plot of 1-AUC vs number of rules (ggplot object).
`stab` plot of stability vs number of rules (ggplot object).

Examples

```
## load sirius
require(sirius)

## prepare data
data <- iris
y <- rep(0, nrow(data))
y[data$Species == 'setosa'] = 1
data$Species <- NULL
```

```
## run cv
cv.grid <- sirius.cv(data, y, nfold = 3, ncv = 2, num.trees = 100)

## plot cv result
plot.error <- sirius.plot.cv(cv.grid)$error
plot(plot.error)
```

sirius.predict	<i>Predict</i>
----------------	----------------

Description

Predictions of a SIRUS model for new observations.

Usage

```
sirius.predict(sirius.m, data.test)
```

Arguments

sirius.m	A SIRUS model generated by sirius.fit.
data.test	Testing data (dataframe of new observations).

Value

Predictions. A vector of the predicted probability of each new observation to be of class 1.

Examples

```
## load sirius
require(sirius)

## prepare data
data <- iris
y <- rep(0, nrow(data))
y[data$Species == 'setosa'] = 1
data$Species <- NULL

#' ## fit sirius
sirius.m <- sirius.fit(data, y)

## predict
predictions <- sirius.predict(sirius.m, data)
```

sirius.print	<i>Print SIRUS</i>
--------------	--------------------

Description

Print the list of rules output by SIRUS.

Usage

```
sirius.print(sirius.m)
```

Arguments

`sirius.m` A SIRUS model generated by `sirius.fit`.

Value

Formatted list of rules.

Examples

```
## load sirius
require(sirius)

## prepare data
data <- iris
y <- rep(0, nrow(data))
y[data$Species == 'setosa'] = 1
data$Species <- NULL

## fit sirius
sirius.m <- sirius.fit(data, y)

## print sirius model
sirius.print(sirius.m)
```

Index

`sirus.cv`, [2](#)
`sirus.fit`, [3](#)
`sirus.plot.cv`, [5](#)
`sirus.predict`, [6](#)
`sirus.print`, [7](#)