Package ‘solitude’

October 14, 2022

Type Package

Title An Implementation of Isolation Forest

Version 1.1.3

Description Isolation forest is anomaly detection method introduced by the paper Isolation based Anomaly Detection (Liu, Ting and Zhou <doi:10.1145/2133360.2133363>).

URL `https://github.com/talegari/solitude`

BugReports `https://github.com/talegari/solitude/issues`

Imports ranger (>= 0.11.0), data.table (>= 1.11.4), igraph (>= 1.2.2), future.apply (>= 0.2.0), R6 (>= 2.4.0), lgr (>= 0.3.4),

Depends R (>= 3.5.0),

Suggests tidyverse, uwot, mlbench, rsample

License GPL-3

Encoding UTF-8

RoxygenNote 7.1.1

NeedsCompilation no

Author Komala Sheshachala Srikanth [aut, cre],
David Zimmermann [ctb]

Maintainer Komala Sheshachala Srikanth <`sri.teach@gmail.com`>

Repository CRAN

Date/Publication 2021-07-29 20:00:02 UTC

R topics documented:

`isolationForest` *Fit an Isolation Forest*

Description

'solitude' class implements the isolation forest method introduced by paper Isolation based Anomaly Detection (Liu, Ting and Zhou <doi:10.1145/2133360.2133363>). The extremely randomized trees (extratrees) required to build the isolation forest is grown using `ranger` function from **ranger** package.

Design

`$new()` initiates a new 'solitude' object. The possible arguments are:

- `sample_size`: (positive integer, default = 256) Number of observations in the dataset to used to build a tree in the forest
- `num_trees`: (positive integer, default = 100) Number of trees to be built in the forest
- `replace`: (boolean, default = FALSE) Whether the sample of observations should be chosen with replacement when sample_size is less than the number of observations in the dataset
- `seed`: (positive integer, default = 101) Random seed for the forest
- `nproc`: (NULL or a positive integer, default: NULL, means use all resources) Number of parallel threads to be used by ranger
- `respect_unordered_factors`: (string, default: "partition")See respect.unordered.factors argument in `ranger`
- `max_depth`: (positive number, default: ceiling(log2(sample_size))) See max.depth argument in `ranger`

`$fit()` fits a isolation forest for the given dataframe or sparse matrix, computes depths of terminal nodes of each tree and stores the anomaly scores and average depth values in `$scores` object as a data.table

`$predict()` returns anomaly scores for a new data as a data.table

Details

- Parallelization: `ranger` is parallelized and by default uses all the resources. This is supported when nproc is set to NULL. The process of obtaining depths of terminal nodes (which is excuted with `$fit()` is called) may be parallelized separately by setting up a **future** backend.

Methods

Public methods:

- `isolationForest$new()`
- `isolationForest$fit()`
- `isolationForest$predict()`
- `isolationForest$clone()`

Method `new()`**:**

Usage:

```
isolationForest$new(
  sample_size = 256,
  num_trees = 100,
  replace = FALSE,
  seed = 101,
  nproc = NULL,
  respect_unordered_factors = NULL,
  max_depth = ceiling(log2(sample_size))
)
```

Method `fit()`**:**

Usage:

```
isolationForest$fit(dataset)
```

Method `predict()`**:**

Usage:

```
isolationForest$predict(data)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
isolationForest$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
## Not run:
library("solitude")
library("tidyverse")
library("mlbench")

data(PimaIndiansDiabetes)
PimaIndiansDiabetes = as_tibble(PimaIndiansDiabetes)
PimaIndiansDiabetes

splitter   = PimaIndiansDiabetes %>%
  select(-diabetes) %>%
  rsample::initial_split(prop = 0.5)
pima_train = rsample::training(splitter)
pima_test  = rsample::testing(splitter)

iso = isolationForest$new()
iso$fit(pima_train)

scores_train = pima_train %>%
  iso$predict() %>%
  arrange(desc(anomaly_score))
```

```
scores_train

umap_train = pima_train %>%
  scale() %>%
  uwot::umap() %>%
  setNames(c("V1", "V2")) %>%
  as_tibble() %>%
  rowid_to_column() %>%
  left_join(scores_train, by = c("rowid" = "id"))

umap_train

umap_train %>%
  ggplot(aes(V1, V2)) +
  geom_point(aes(size = anomaly_score))

scores_test = pima_test %>%
  iso$predict() %>%
  arrange(desc(anomaly_score))

scores_test

## End(Not run)
```

`is_integerish` *Check for a single integer*

Description

for a single integer

Usage

```
is_integerish(x)
```

Arguments

`x`	input

Value

TRUE or FALSE

Examples

```
## Not run: is_integerish(1)
```

`solitude` *An Implementation of Isolation Forest*

Description

Isolation forest is an anomaly detection method introduced by the paper Isolation based Anomaly Detection (Liu, Ting and Zhou <doi:10.1145/2133360.2133363>)

Author(s)

Srikanth Komala Sheshachala

See Also

Useful links:

- `https://github.com/talegari/solitude`
- Report bugs at `https://github.com/talegari/solitude/issues`

`terminalNodesDepth` *Depth of each terminal node of all trees in a ranger model*

Description

Depth of each terminal node of all trees in a ranger model is returned as a three column tibble with column names: 'id_tree', 'id_node', 'depth'. Note that root node has the node_id = 0.

Usage

```
terminalNodesDepth(model)
```

Arguments

`model`	A ranger model

Details

This function may be parallelized using a future backend.

Value

A tibble with three columns: 'id_tree', 'id_node', 'depth'.

Examples

```
rf = ranger::ranger(Species ~ ., data = iris, num.trees = 100)
terminalNodesDepth(rf)
```

`terminalNodesDepthPerTree` *Depth of each terminal node of a single tree in a ranger model*

Description

Depth of each terminal node of a single tree in a ranger model. Note that root node has the id_node = 0.

Usage

```
terminalNodesDepthPerTree(treelike)
```

Arguments

`treelike`	Output of ‘ranger::treeInfo‘

Value

data.table with two columns: id_node and depth

Examples

```
## Not run:
  rf = ranger::ranger(Species ~ ., data = iris)
  terminalNodesDepthPerTree(ranger::treeInfo(rf, 1))

## End(Not run)
```

Index