Package 'arenar'

July 27, 2020

Title Arena for the Exploration and Comparison of any ML Models

Version 0.1.8

Description Generates data for challenging machine learning models in 'Arena' <https://arena.drwhy.ai> - an interactive web application. You can start the server with XAI (Explainable Artificial Intelligence) plots to be generated on-demand or precalculate and auto-upload data file beside shareable 'Arena' URL.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.0

Depends R (>= 3.6)

Imports ingredients, iBreakDown, gistr, jsonlite, plumber, parallel, utils, stats, methods, auditor, DALEX (>= 1.3.0)

Suggests testthat, knitr, rmarkdown, dplyr, pkgdown, covr, ranger

VignetteBuilder knitr

URL `https://arenar.drwhy.ai`, `https://github.com/ModelOriented/ArenaR`

BugReports `https://github.com/ModelOriented/ArenaR/issues`

NeedsCompilation no

Author Piotr Piątyszek [aut, cre], Przemyslaw Biecek [aut] (<https://orcid.org/0000-0001-8423-1823>)

Maintainer Piotr Piątyszek `<piotrp@wektor.xyz>`

Repository CRAN

Date/Publication 2020-07-27 13:10:02 UTC

R topics documented:

`create_arena` *Creates arena object*

Description

Creates object with class `arena_live` or `arena_static` depending on the first argument. This method is always first in `arenar` workflow and you should specify all plots' parameters there.

Usage

```
create_arena(
  live = FALSE,
  N = 500,
  fi_N = NULL,
  fi_B = 10,
  grid_points = 101,
  shap_B = 10,
  funnel_nbins = 5,
  funnel_cutoff = 0.01,
  funnel_factor_threshold = 7,
  cl = NULL
)
```

Arguments

`live`	Defines if arena should start live server or generate static json
`N`	number of observations used to calculate dependence profiles
`fi_N`	number of observations used in feature importance
`fi_B`	Number of permutation rounds to perform each variable in feature importance
`grid_points`	number of points for profile
`shap_B`	Numer of random paths in SHAP
`funnel_nbins`	Number of partitions for numeric columns for funnel plot
`funnel_cutoff`	Threshold for categorical data. Entries less frequent than specified value will be merged into one category in funnel plot.
`funnel_factor_threshold`	Numeric columns with lower number of unique values than value of this parameter will be treated as factors in funnel plot.
`cl`	Cluster used to run parallel computations (Do not work in live Arena)

Value

Empty `arena_static` or `arena_live` class object.
`arena_static`:

- explainer List of used explainers
- observations_batches List of data frames added as observations
- params Plots' parameters
- plots_data List of generated data for plots

`arena_live`:

- explainer List of used explainers
- observations_batches List of data frames added as observations
- params Plots' parameters
- timestamp Timestamp of last modification

Examples

```
library("DALEX")
library("arenar")
library("dplyr", quietly=TRUE, warn.conflicts = FALSE)
# create a model
model <- glm(m2.price ~ ., data=apartments)
# create a DALEX explainer
explainer <- DALEX::explain(model, data=apartments, y=apartments$m2.price)
# prepare observations to be explained
observations <- apartments[1:3, ]
# rownames are used as labels for each observation
rownames(observations) <- paste0(observations$construction.year, "-", observations$surface, "m2")
# generate static arena for one model and 3 observations
```

```
arena <- create_arena(live=FALSE) %>% push_model(explainer) %>% push_observations(observations)
print(arena)
if (interactive()) upload_arena(arena)
```

`funnel_measure` *Internal function for calculating data for funnel plot*

Description

This is simplified version of `DALEXtra::funnel_measure`

Usage

```
funnel_measure(
  explainer,
  measure_function = NULL,
  nbins = 5,
  cutoff = 0.01,
  cutoff_name = "Other",
  factor_conversion_threshold = 7
)
```

Arguments

`explainer`	Explainer created using `DALEX::explain`
`measure_function`	measure function that calculates performance of model based on true observation and prediction. Order of parameters is important and should be (y, y_hat). The measure calculated by the function should have the property that lower score value indicates better model. If NULL, RMSE will be used for regression, one minus auc for classification and crossentropy for multiclass classification.
`nbins`	Number of qunatiles (partition points) for numeric columns. In case when more than one qunatile have the same value, there will be less partition points.
`cutoff`	Threshold for categorical data. Entries less frequent than specified value will be merged into one category.
`cutoff_name`	Name for new category that arised after merging entries less frequent than `cutoff`
`factor_conversion_threshold`	Numeric columns with lower number of unique values than value of this parameter will be treated as factors

Value

Data frame with columns

- Variable Name of splited variable
- Measure Loss value for subset
- Label Label for variable's values subset

`get_accumulated_dependence` *Internal function for calculating Accumulated Dependence*

Description

Internal function for calculating Accumulated Dependence

Usage

```
get_accumulated_dependence(explainer, variable, params)
```

Arguments

`explainer`	Explainer created using `DALEX::explain`
`variable`	Name of variable
`params`	Params from arena object

Value

Plot data in Arena's format

`get_break_down` *Internal function for calculating Break Down*

Description

Internal function for calculating Break Down

Usage

```
get_break_down(explainer, observation, params)
```

Arguments

`explainer`	Explainer created using `DALEX::explain`
`observation`	One row data frame observation
`params`	Params from arena object

Value

Plot data in Arena's format

get_ceteris_paribus *Internal function for calculating Ceteris Paribus*

Description

Internal function for calculating Ceteris Paribus

Usage

```
get_ceteris_paribus(explainer, observation, variable, params)
```

Arguments

`explainer`	Explainer created using `DALEX::explain`
`observation`	One row data frame observation
`variable`	Name of variable
`params`	Params from arena object

Value

Plot data in Arena's format

get_feature_importance *Internal function for calculating feature importance*

Description

Internal function for calculating feature importance

Usage

```
get_feature_importance(explainer, vars, params)
```

Arguments

`explainer`	Explainer created using `DALEX::explain`
`vars`	Variables names for which feature importance should be calculated
`params`	Params from arena object

Value

Plot data in Arena's format

`get_funnel_measure` *Internal function for calculating funnel measure*

Description

Internal function for calculating funnel measure

Usage

```
get_funnel_measure(explainer, params)
```

Arguments

`explainer`	Explainer created using `DALEX::explain`
`params`	Params from arena object

Value

Plot data in Arena's format

`get_global_plots` *Internal function for calculating global plots*

Description

Function runs all plot generating methods for given explainer

Usage

```
get_global_plots(explainer, params)
```

Arguments

`explainer`	Explainer created using `DALEX::explain`
`params`	Params from arena object

Value

list of generated plots' data

`get_json_structure` *Prepare object ready to change into json*

Description

Function converts object with class `arena_live` or `arena_static` to object with structure accepted by Arena. See list of schemas.

Usage

```
get_json_structure(arena)
```

Arguments

`arena`	live or static arena object

Value

Object for direct conversion into json

`get_local_plots` *Internal function for calculating local plots for all observations*

Description

Function runs all plot generating methods for given observations

Usage

```
get_local_plots(explainer, observations, params)
```

Arguments

`explainer`	Explainer created using `DALEX::explain`
`observations`	Data frame of observations
`params`	Params from arena object

Value

list of generated plots' data

`get_metrics` *Internal function for calculating model performance metrics*

Description

Internal function for calculating model performance metrics

Usage

```
get_metrics(explainer, params)
```

Arguments

`explainer`	Explainer created using `DALEX::explain`
`params`	Params from arena object

Value

Plot data in Arena's format

`get_observations_list` *Generates list of rownames of each observation from each batch*

Description

Generates list of rownames of each observation from each batch

Usage

```
get_observations_list(arena)
```

Arguments

`arena`	live or static arena object

Value

list of observations' names

`get_partial_dependence` *Internal function for calculating Partial Dependence*

Description

Internal function for calculating Partial Dependence

Usage

```
get_partial_dependence(explainer, variable, params)
```

Arguments

`explainer`	Explainer created using `DALEX::explain`
`variable`	Name of variable
`params`	Params from arena object

Value

Plot data in Arena's format

`get_rec` *Internal function for calculating regression error characteristic*

Description

Internal function for calculating regression error characteristic

Usage

```
get_rec(explainer, params)
```

Arguments

`explainer`	Explainer created using `DALEX::explain`
`params`	Params from arena object

Value

Plot data in Arena's format

`get_roc` *Internal function for calculating receiver operating curve*

Description

Internal function for calculating receiver operating curve

Usage

```
get_roc(explainer, params)
```

Arguments

`explainer`	Explainer created using `DALEX::explain`
`params`	Params from arena object

Value

Plot data in Arena's format

`get_shap_values` *Internal function for calculating Shapley Values*

Description

Internal function for calculating Shapley Values

Usage

```
get_shap_values(explainer, observation, params)
```

Arguments

`explainer`	Explainer created using `DALEX::explain`
`observation`	One row data frame observation to calculate Shapley Values
`params`	Params from arena object

Value

Plot data in Arena's format

`get_variables_list` *Generates list of unique variables(without target) from each explainer*

Description

Generates list of unique variables(without target) from each explainer

Usage

```
get_variables_list(arena)
```

Arguments

`arena`	live or static arena object

Value

list of variables' names

`print.arena_live` *Prints live arena summary*

Description

Prints live arena summary

Usage

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

Arguments

`x`	`arena_live` object
`...`	other parameters

Value

None

Examples

```
library("DALEX")
library("arenar")
library("dplyr", quietly=TRUE, warn.conflicts = FALSE)
# create a model
model <- glm(m2.price ~ ., data=apartments)
# create a DALEX explainer
explainer <- DALEX::explain(model, data=apartments, y=apartments$m2.price)
# prepare observations to be explained
observations <- apartments[1:30, ]
# rownames are used as labels for each observation
rownames(observations) <- paste0(observations$construction.year, "-", observations$surface, "m2")
# generate live arena for one model and 30 observations
arena <- create_arena(live=TRUE) %>% push_model(explainer) %>% push_observations(observations)
# print summary
print(arena)
```

`print.arena_static` *Prints static arena summary*

Description

Prints static arena summary

Usage

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

Arguments

`x`	`arena_static` object
`...`	other parameters

Value

None

Examples

```
library("DALEX")
library("arenar")
library("dplyr", quietly=TRUE, warn.conflicts = FALSE)
# create a model
model <- glm(m2.price ~ ., data=apartments)
# create a DALEX explainer
explainer <- DALEX::explain(model, data=apartments, y=apartments$m2.price)
# prepare observations to be explained
observations <- apartments[1:3, ]
```

```
# rownames are used as labels for each observation
rownames(observations) <- paste0(observations$construction.year, "-", observations$surface, "m2")
# generate static arena for one model and 3 observations
arena <- create_arena(live=FALSE) %>% push_model(explainer) %>% push_observations(observations)
# print summary
print(arena)
```

push_model *Adds model to arena*

Description

If arena is static it will start calculations for all already pushed observations and global plots. If arena is live, then plots will be calculated on demand, after calling `arena_run`.

Usage

```
push_model(arena, explainer)
```

Arguments

`arena`	live or static arena object
`explainer`	Explainer created using `DALEX::explain`

Value

Updated arena object

Examples

```
library("DALEX")
library("arenar")
library("dplyr", quietly=TRUE, warn.conflicts = FALSE)
# create first model
model1 <- glm(m2.price ~ ., data=apartments, family=gaussian)
# create a DALEX explainer
explainer1 <- DALEX::explain(model1, data=apartments, y=apartments$m2.price, label="GLM gaussian")
# create live arena with only one model
arena <- create_arena(live=TRUE) %>% push_model(explainer1)
print(arena)
# create and add next model
model2 <- glm(m2.price ~ ., data=apartments, family=Gamma)
explainer2 <- DALEX::explain(model2, data=apartments, y=apartments$m2.price, label="GLM gamma")
arena <- arena %>% push_model(explainer2)
print(arena)
```

`push_observations` *Adds new observations to arena*

Description

If arena is static it will start calculations for all already pushed models. If arena is live, then plots will be calculated on demand, after calling `arena_run`.

Usage

```
push_observations(arena, observations)
```

Arguments

`arena`	live or static arena object
`observations`	data frame of new observations

Value

Updated arena object

`run_server` *Run server providing data for live Arena*

Description

By default function opens browser with new arena session. Appending data to already existing session is also possible using argument `append_data`

Usage

```
run_server(
  arena,
  port = 8181,
  host = "127.0.0.1",
  open_browser = TRUE,
  append_data = FALSE,
  arena_url = "https://arena.drwhy.ai/"
)
```

Arguments

`arena`	Live arena object
`port`	server port
`host`	server ip address (hostnames do not work yet)
`open_browser`	Whether to open browser with new session
`append_data`	Whether to append data to already existing session
`arena_url`	URL of Arena dashboard instance

Value

not modified arena object

Examples

```
library("DALEX")
library("arenar")
library("dplyr", quietly=TRUE, warn.conflicts = FALSE)
# create a model
model <- glm(m2.price ~ ., data=apartments)
# create a DALEX explainer
explainer <- DALEX::explain(model, data=apartments, y=apartments$m2.price)
# generate live arena for one model and all data as observations
arena <- create_arena(live=TRUE) %>% push_model(explainer) %>% push_observations(apartments)
# run the server
if (interactive()) run_server(arena, port=1234)
```

`split_multiclass_explainer` *Splits multiclass explainer into multiple classification explainers*

Description

Splits multiclass explainer into multiple classification explainers

Usage

```
split_multiclass_explainer(explainer)
```

Arguments

`explainer`	Multiclass explainer created using `DALEX::explain`

Value

list of explainers

`upload_arena` *Upload generated json file from static arena*

Description

By default function opens browser with new arena session. Appending data to already existing session is also possible using argument `append_data`

Usage

```
upload_arena(
  arena,
  open_browser = TRUE,
  append_data = FALSE,
  arena_url = "https://arena.drwhy.ai/",
  pretty = FALSE
)
```

Arguments

`arena`	Static arena object
`open_browser`	Whether to open browser with new session
`append_data`	Whether to append data to already existing session
`arena_url`	URL of Arena dashboard instance
`pretty`	whether to generate pretty and easier to debug JSON

Value

not modified arena object

`validate_new_model` *Checks if it is safe do add a new model to the arena object*

Description

Function checks if explainer's label is not already used call stop if there is at least one conflict.

Usage

```
validate_new_model(arena, explainer)
```

Arguments

`arena`	live or static arena object
`explainer`	Explainer created using `DALEX::explain`

Value

None

`validate_new_observations` *Checks if it is safe do add new observations to the arena object*

Description

Function checks if rownames are not already used and call stop if there is at least one conflict.

Usage

```
validate_new_observations(arena, observations)
```

Arguments

`arena`	live or static arena object
`observations`	data frame of new observations

Value

None

Index