Package ‘arenar’

October 12, 2022

Title Arena for the Exploration and Comparison of any ML Models

Version 0.2.0

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 7.1.1

Depends R (>= 3.6)

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

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

VignetteBuilder knitr


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

NeedsCompilation no

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

Maintainer Piotr Piętyszek <piotrp@wektor.xyz>

Repository CRAN

Date/Publication 2020-10-01 08:00:06 UTC
### R topics documented:

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>calculate_subsets_performance</td>
<td>3</td>
</tr>
<tr>
<td>create_arena</td>
<td>4</td>
</tr>
<tr>
<td>get_accumulated_dependence</td>
<td>6</td>
</tr>
<tr>
<td>get_attributes</td>
<td>6</td>
</tr>
<tr>
<td>get_break_down</td>
<td>7</td>
</tr>
<tr>
<td>get_ceteris_paribus</td>
<td>7</td>
</tr>
<tr>
<td>get_datasets_list</td>
<td>8</td>
</tr>
<tr>
<td>get_dataset_attributes</td>
<td>8</td>
</tr>
<tr>
<td>get_dataset_plots</td>
<td>9</td>
</tr>
<tr>
<td>get_fairness</td>
<td>9</td>
</tr>
<tr>
<td>get_feature_importance</td>
<td>10</td>
</tr>
<tr>
<td>get_funnel_measure</td>
<td>10</td>
</tr>
<tr>
<td>get_global_plots</td>
<td>11</td>
</tr>
<tr>
<td>get_json_structure</td>
<td>11</td>
</tr>
<tr>
<td>get_local_plots</td>
<td>12</td>
</tr>
<tr>
<td>get_message_output</td>
<td>12</td>
</tr>
<tr>
<td>get_metrics</td>
<td>13</td>
</tr>
<tr>
<td>get_model_attributes</td>
<td>13</td>
</tr>
<tr>
<td>get_observations_list</td>
<td>14</td>
</tr>
<tr>
<td>get_observation_attributes</td>
<td>14</td>
</tr>
<tr>
<td>get_partial_dependence</td>
<td>15</td>
</tr>
<tr>
<td>get_rec</td>
<td>15</td>
</tr>
<tr>
<td>get_roc</td>
<td>16</td>
</tr>
<tr>
<td>get_shap_values</td>
<td>16</td>
</tr>
<tr>
<td>get_subsets_performance</td>
<td>17</td>
</tr>
<tr>
<td>get_variables_list</td>
<td>17</td>
</tr>
<tr>
<td>get_variable_against_another</td>
<td>18</td>
</tr>
<tr>
<td>get_variable_attributes</td>
<td>18</td>
</tr>
<tr>
<td>get_variable_distribution</td>
<td>19</td>
</tr>
<tr>
<td>print.arena_live</td>
<td>19</td>
</tr>
<tr>
<td>print.arena_static</td>
<td>20</td>
</tr>
<tr>
<td>push_dataset</td>
<td>21</td>
</tr>
<tr>
<td>push_model</td>
<td>22</td>
</tr>
<tr>
<td>push_observations</td>
<td>23</td>
</tr>
<tr>
<td>run_server</td>
<td>23</td>
</tr>
<tr>
<td>save_arena</td>
<td>24</td>
</tr>
<tr>
<td>split_multiclass_explainer</td>
<td>25</td>
</tr>
<tr>
<td>truncate_vector</td>
<td>25</td>
</tr>
<tr>
<td>upload_arena</td>
<td>26</td>
</tr>
<tr>
<td>validate_new_dataset</td>
<td>26</td>
</tr>
<tr>
<td>validate_new_model</td>
<td>27</td>
</tr>
<tr>
<td>validate_new_observations</td>
<td>27</td>
</tr>
</tbody>
</table>

**Index**

29
calculate_subsets_performance

Internal function for calculating data for funnel plot

Description

This is modified version of DALEXtra::funnel_measure

Usage

calculate_subsets_performance(
  explainer,
  score_functions = list(),
  nbins = 5,
  cutoff = 0.01,
  cutoff_name = "Other",
  factor_conversion_threshold = 7
)

Arguments

explainer Explainer created using DALEX::explain
score_functions Named list of functions named score_* from auditor package
nbins Number of quantiles (partition points) for numeric columns. In case when more than one quantile 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 splitted variable
- Label Label for variable’s values subset

and one column for each score function with returned score
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,
  fairness_cutoffs = seq(0.05, 0.95, 0.05),
  max_points_number = 150,
  distribution_bins = seq(5, 40, 5),
  enable_attributes = TRUE,
  enable_custom_params = TRUE,
  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
  Number 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.

fairness_cutoffs
  vector of available cutoff levels for fairness panel
create_arena

max_points_number
maximum size of sample to plot scatter plots in variable against another panel
distribution_bins
vector of available bins count for histogram
enable_attributes
Switch for generating attributes of observations and variables. It is required for custom params. Attributes can increase size of static Arena.
enable_custom_params
Switch to allowing user to modify observations and generate plots for them.
c1
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)
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_attributes

Returns attributes for all params

Description

When param_type is not NULL, then function returns list of objects. Each object represents one of available attribute for specified param type. Field name is attribute name and field values is mapped list of available params to list of value of this attribute for that param. When param_type is NULL, then function returns list with keys for each param type and values are lists described above.

Usage

get_attributes(arena, param_type = NULL)

Arguments

arena live or static arena object
param_type Type of param. One of
• model
• variable
• dataset
• observation
**get_break_down**

**Value**

List of attributes or named list of lists of attributes for each param type.

---

**get_break_down**  
*Internal function for calculating Break Down*

**Description**

Internal function for calculating Break Down

**Usage**

```r
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**

```r
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_datasets_list

Generates list of datasets’ labels

Description
Generates list of datasets’ labels

Usage
get_datasets_list(arena)

Arguments
arena live or static arena object

Value
list of datasets’ labels

get_dataset_attributes

Generates list with attributes of a dataset

Description
Generates list with attributes of a dataset

Usage
get_dataset_attributes(arena, dataset)

Arguments
arena live or static arena object
dataset List with following elements
• dataset Data frame
• target Name of one column from data frame that is used as target variable
• label Label for dataset to be displayed in Arena
• variables vector of column names from data frame without target

Value
simple list with attributes of given dataset
**get_dataset_plots**  
*Internal function for calculating exploratory data analysis plots*

**Description**
Function runs all plot generating methods for given dataset

**Usage**

get_dataset_plots(dataset, params)

**Arguments**
- **dataset**
  - List with following elements
    - dataset Data frame
    - target Name of one column from data frame that is used as target variable
    - label Label for dataset to be displayed in Arena
    - variables vector of column names from data frame without target

- **params**
  - Params from arena object

**Value**
list of generated plots’ data

**get_fairness**  
*Internal function for calculating fairness*

**Description**
Internal function for calculating fairness

**Usage**

get_fairness(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_feature_importance

*Internal function for calculating feature importance*

**Description**

Internal function for calculating feature importance

**Usage**

```r
get_feature_importance(explainer, params)
```

**Arguments**

- `explainer` Explainer created using `DALEX::explain`
- `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**

```r
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

global_plots(explainer, params)

Arguments

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

Value

list of generated plots’ data

global_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**

```r
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_message_output  
*Internal function for returning message as plot data*

**Description**

This method modify existing plot’s data in Arena’s format to show message instead of chart.

**Usage**

```r
get_message_output(output, type, msg)
```

**Arguments**

- `output`: existing plot data to be overwritten
- `type`: type of message “info” or “error”
- `msg`: message to be displayed

**Value**

Plot data in Arena’s format
**get_metrics**

*Internal function for calculating model performance metrics*

**Description**

Internal function for calculating model performance metrics

**Usage**

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

**Arguments**

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

**Value**

Plot data in Arena’s format

---

**get_model_attributes**

*Generates list with attributes of a model*

**Description**

Generates list with attributes of a model

**Usage**

```r
get_model_attributes(arena, explainer)
```

**Arguments**

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

**Value**

simple list with attributes of given model
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_observation_attributes  Generates list with attributes of an observation

Description
Generates list with attributes of an observation

Usage
get_observation_attributes(arena, observation)

Arguments
arena  live or static arena object
observation  One row data frame observation

Value
simple list with attributes of given observation
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_subsets_performance

Internal function for calculating subset performance

Description

Internal function for calculating subset performance

Usage

get_subsets_performance(explainer, params)

Arguments

explainer | Explainer created using DALEX::explain
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 and dataset

Description

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

Usage

get_variables_list(arena)

Arguments

arena | live or static arena object

Value

list of variables’ names
get_variable_against_another

Internal function for variable against another plot

Description
Internal function for variable against another plot

Usage
get_variable_against_another(dataset, variable, params)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dataset</td>
<td>List with following elements</td>
</tr>
<tr>
<td></td>
<td>• dataset Data frame</td>
</tr>
<tr>
<td></td>
<td>• target Name of one column from data frame that is used as target variable</td>
</tr>
<tr>
<td></td>
<td>• label Label for dataset to be displayed in Arena</td>
</tr>
<tr>
<td></td>
<td>• variables vector of column names from data frame without target</td>
</tr>
<tr>
<td>variable</td>
<td>Name of primary variable</td>
</tr>
<tr>
<td>params</td>
<td>Params from arena object</td>
</tr>
</tbody>
</table>

Value
Plot data in Arena’s format

get_variable_attributes

Generates list with attributes of an variable

Description
Generates list with attributes of an variable

Usage
get_variable_attributes(arena, variable)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>arena</td>
<td>live or static arena object</td>
</tr>
<tr>
<td>variable</td>
<td>Name of variable</td>
</tr>
</tbody>
</table>

Value
simple list with attributes of given variable
**get_variable_distribution**

*Internal function for variable distribution*

**Description**

Internal function for variable distribution

**Usage**

`get_variable_distribution(dataset, variable, params)`

**Arguments**

- **dataset**
  - List with following elements
  - `dataset` Data frame
  - `target` Name of one column from data frame that is used as target variable
  - `label` Label for dataset to be displayed in Arena
  - `variables` vector of column names from data frame without target

- **variable**
  - Name of variable

- **params**
  - Params from arena object

**Value**

Plot data in Arena’s format

---

**print.arena_live**

*Prints live arena summary*

**Description**

Prints live arena summary

**Usage**

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

**Arguments**

- **x**
  - arena_live object

- **other parameters**
print.arena_static

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
push_dataset

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_dataset

*Adds new datasets to Arena*

Description

Adds data frame to create exploratory data analysis plots

Usage

```
push_dataset(arena, dataset, target, label)
```

Arguments

- **arena**: live or static arena object
- **dataset**: data frame used for EDA plots
- **target**: name of target variable
- **label**: label of dataset

Value

Updated arena object

Examples

```
library("DALEX")
library("arenar")
library("dplyr", quietly=TRUE, warn.conflicts = FALSE)
# create live arena with only one dataset
apartments <- DALEX::apartments
arena <- create_arena(live=TRUE) %>% push_dataset(apartments, "m2.price", "apartment")
print(arena)
```
# add another dataset
HR <- DALEX::HR
arena <- arena %>% push_dataset(HR, "status", "HR")
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

```r
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**

```r
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**

```r
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

```r
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)
```

Description

Save generated json file from static arena

Usage

```r
save_arena(arena, filename = "data.json", pretty = FALSE)
```
**split_multiclass_explainer**

*Splits multiclass explainer into multiple classification explainers*

**Description**

Splits multiclass explainer into multiple classification explainers

**Usage**

```r
split_multiclass_explainer(explainer)
```

**Arguments**

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

**Value**

list of explainers

---

**truncate_vector**  
*Internal function for pretty truncationg params list*

**Description**

Internal function for pretty truncationg params list

**Usage**

```r
truncate_vector(vec, size = 6)
```

**Arguments**

- `vec`  
  vector to be truncated
- `size`  
  elements with index greater than size will be truncated

**Value**

string with collapsed and truncated input vector
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

```r
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_dataset

Checks if it is safe do add new dataset to the arena object

Description

Checks if it is safe do add new dataset to the arena object

Usage

```r
validate_new_dataset(arena, dataset, target, label)
```
validate_new_model

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>arena</td>
<td>live or static arena object</td>
</tr>
<tr>
<td>dataset</td>
<td>data frame for data analysis</td>
</tr>
<tr>
<td>target</td>
<td>name of target variable</td>
</tr>
<tr>
<td>label</td>
<td>name of dataset</td>
</tr>
</tbody>
</table>

Value

None

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

Description

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

Usage

validate_new_model(arena, explainer)

Argument

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>arena</td>
<td>live or static arena object</td>
</tr>
<tr>
<td>explainer</td>
<td>Explainer created using DALEX::explain</td>
</tr>
</tbody>
</table>

Value

None

validate_new_observations

Checks if it is safe to 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)
validate_new_observations

Arguments

- arena: live or static arena object
- observations: data frame of new observations

Value

None
Index

calculate_subsets_performance, 3
create_arena, 4
get_accumulated_dependence, 6
get_attributes, 6
get_break_down, 7
get_ceteris_paribus, 7
get_dataset_attributes, 8
get_dataset_plots, 9
get_datasets_list, 8
get_fairness, 9
get_feature_importance, 10
get_funnel_measure, 10
get_global_plots, 11
get_json_structure, 11
get_local_plots, 12
get_message_output, 12
get_metrics, 13
get_model_attributes, 13
get_observation_attributes, 14
get_observations_list, 14
get_partial_dependence, 15
get_rec, 15
get_roc, 16
get_shap_values, 16
get_subsets_performance, 17
get_variable_against_another, 18
get_variable_attributes, 18
get_variable_distribution, 19
get_variables_list, 17
print.arena_live, 19
print.arena_static, 20
push_dataset, 21
push_model, 22
push_observations, 23
run_server, 23
save_arena, 24
split_multiclass_explainer, 25
truncate_vector, 25
upload_arena, 26
validate_new_dataset, 26
validate_new_model, 27
validate_new_observations, 27