

# Package ‘arenar’

October 1, 2020

**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

**URL** <https://arenar.drwhy.ai>, <https://github.com/ModelOriented/ArenaR>

**BugReports** <https://github.com/ModelOriented/ArenaR/issues>

**NeedsCompilation** no

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

**Maintainer** Piotr Piątyśzek <[piotrp@wektor.xyz](mailto:piotrp@wektor.xyz)>

**Repository** CRAN

**Date/Publication** 2020-10-01 08:00:06 UTC

**R topics documented:**

calculate_subsets_performance . . . . .	3
create_arena . . . . .	4
get_accumulated_dependence . . . . .	6
get_attributes . . . . .	6
get_break_down . . . . .	7
get_ceteris_paribus . . . . .	7
get_datasets_list . . . . .	8
get_dataset_attributes . . . . .	8
get_dataset_plots . . . . .	9
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_observations_list . . . . .	14
get_observation_attributes . . . . .	14
get_partial_dependence . . . . .	15
get_rec . . . . .	15
get_roc . . . . .	16
get_shap_values . . . . .	16
get_subsets_performance . . . . .	17
get_variables_list . . . . .	17
get_variable_against_another . . . . .	18
get_variable_attributes . . . . .	18
get_variable_distribution . . . . .	19
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

---

`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

<code>explainer</code>	Explainer created using DALEX::explain
<code>score_functions</code>	Named list of functions named <code>score_*</code> from auditor package
<code>nbins</code>	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.
<code>cutoff</code>	Threshold for categorical data. Entries less frequent than specified value will be merged into one category.
<code>cutoff_name</code>	Name for new category that arised after merging entries less frequent than cutoff
<code>factor_conversion_threshold</code>	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
- Label Label for variable's values subset

and one column for each score function with returned score

---

create_arena	<i>Creates arena object</i>
--------------	-----------------------------

---

### 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

<code>live</code>	Defines if arena should start live server or generate static json
<code>N</code>	number of observations used to calculate dependence profiles
<code>fi_N</code>	number of observations used in feature importance
<code>fi_B</code>	Number of permutation rounds to perform each variable in feature importance
<code>grid_points</code>	number of points for profile
<code>shap_B</code>	Numer of random paths in SHAP
<code>funnel_nbins</code>	Number of partitions for numeric columns for funnel plot
<code>funnel_cutoff</code>	Threshold for categorical data. Entries less frequent than specified value will be merged into one category in funnel plot.
<code>funnel_factor_threshold</code>	Numeric columns with lower number of unique values than value of this parameter will be treated as factors in funnel plot.
<code>fairness_cutoffs</code>	vector of available cutoff levels for fairness panel

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.
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)
```

---

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 <ul style="list-style-type: none"> <li>• model</li> <li>• variable</li> <li>• dataset</li> <li>• observation</li> </ul>

**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**

```
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\_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 <ul style="list-style-type: none"><li>• dataset Data frame</li><li>• target Name of one column from data frame that is used as target variable</li><li>• label Label for dataset to be displayed in Arena</li><li>• variables vector of column names from data frame without target</li></ul>
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**

```
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**

```
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	<i>Internal function for calculating global plots</i>
------------------	---

---

**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	<i>Prepare object ready to change into json</i>
--------------------	---

---

**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	<i>Internal function for calculating local plots for all observations</i>
-----------------	---

---

**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_message_output	<i>Internal function for returning message as plot data</i>
--------------------	---

---

**Description**

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

**Usage**

```
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	<i>Internal function for calculating model performance metrics</i>
-------------	--

---

**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_model_attributes	<i>Generates list with attributes of a model</i>
----------------------	--

---

**Description**

Generates list with attributes of a model

**Usage**

```
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**

<code>explainer</code>	Explainer created using DALEX::explain
<code>variable</code>	Name of variable
<code>params</code>	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**

<code>explainer</code>	Explainer created using DALEX::explain
<code>params</code>	Params from arena object

**Value**

Plot data in Arena's format

---

get_roc	<i>Internal function for calculating receiver operating curve</i>
---------	---

---

**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	<i>Internal function for calculating Shapley Values</i>
-----------------	---

---

**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

dataset	List with following elements <ul style="list-style-type: none"> <li>• dataset Data frame</li> <li>• target Name of one column from data frame that is used as target variable</li> <li>• label Label for dataset to be displayed in Arena</li> <li>• variables vector of column names from data frame without target</li> </ul>
variable	Name of primary variable
params	Params from arena object

### 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

arena	live or static arena object
variable	Name of variable

### 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 <ul style="list-style-type: none"><li>• dataset Data frame</li><li>• target Name of one column from data frame that is used as target variable</li><li>• label Label for dataset to be displayed in Arena</li><li>• variables vector of column names from data frame without target</li></ul>
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**

```
## 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\_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

```
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	<i>Adds new observations to arena</i>
-------------------	---------------------------------------

---

**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	<i>Run server providing data for live Arena</i>
------------	---

---

**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)
```

---

save\_arena

*Save generated json file from static arena*

---

**Description**

Save generated json file from static arena

**Usage**

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

**Arguments**

arena	Static arena object
filename	Name of output file
pretty	whether to generate pretty and easier to debug JSON

**Value**

not modified arena object



---

`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

---

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

---

**Description**

Internal function for pretty truncationg params list

**Usage**

```
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	<i>Upload generated json file from static arena</i>
--------------	---

---

### 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

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

### Value

not modified arena object

---

validate_new_dataset	<i>Checks if it is safe do add new dataset to the arena object</i>
----------------------	--

---

### Description

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

### Usage

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

**Arguments**

arena	live or static arena object
dataset	data frame for data analysis
target	name of target variable
label	name of dataset

**Value**

None

---

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

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](#)