Package 'mlflow'

February 3, 2026

Type Package

Title Interface to 'MLflow'

Version 3.9.0

Description R interface to 'MLflow', open source platform for the complete machine learning life cycle, see <https://mlflow.org/>. This package supports installing 'MLflow', tracking experiments, creating and running projects, and saving and serving models.

License Apache License 2.0

URL https://github.com/mlflow/mlflow

BugReports https://github.com/mlflow/mlflow/issues

Depends R (>= 3.3.0)

Imports base64enc, fs, git2r, glue, httpuv, httr, ini, jsonlite, openssl, processx, purrr, rlang (>= 0.2.0), swagger, tibble (>= 2.0.0), withr, yaml, zeallot

Suggests carrier, covr, h2o, keras, lintr, sparklyr, stringi, testthat (>= 2.0.0), reticulate, xgboost

Encoding UTF-8

RoxygenNote 7.1.2

Collate 'cast-utils.R' 'cli.R' 'databricks-utils.R' 'globals.R' 'imports.R' 'logging.R' 'mlflow-package.R' 'model-crate.R' 'model-python.R' 'model.R' 'model-utils.R' 'model-h2o.R' 'model-keras.R' 'model-registry.R' 'model-serve.R' 'model-swagger.R' 'model-xgboost.R' 'project-param.R' 'project-run.R' 'project-source.R' 'python.R' 'tracking-client.R' 'tracking-experiments.R' 'tracking-observer.R' 'tracking-globals.R' 'tracking-rest.R' 'tracking-runs.R' 'tracking-server.R' 'tracking-ui.R' 'tracking-utils.R'

NeedsCompilation no

Author Ben Wilson [aut, cre],
Matei Zaharia [aut],
Javier Luraschi [aut],
Kevin Kuo [aut] (ORCID: <https://orcid.org/0000-0001-7803-7901>),
RStudio [cph]

Maintainer Ben Wilson <benjamin.wilson@databricks.com>

Repository CRAN

Date/Publication 2026-02-03 18:40:02 UTC

Contents

`build_context_tags_from_databricks_job_info`
Get information from a Databricks job execution context

Description

Parses the data from a job execution context when running on Databricks in a non-interactive mode. This function extracts relevant data that MLflow needs in order to properly utilize the MLflow APIs from this context.

Usage

```
build_context_tags_from_databricks_job_info(job_info)
```

Arguments

`job_info`	The job-related metadata from a running Databricks job

Value

A list of tags to be set by the run context when creating MLflow runs in the current Databricks Job environment

build_context_tags_from_databricks_notebook_info *Get information from Databricks Notebook environment*

Description

Retrieves the notebook id, path, url, name, version, and type from the Databricks Notebook execution environment and sets them to a list to be used for setting the configured environment for executing an MLflow run in R from Databricks.

Usage

```
build_context_tags_from_databricks_notebook_info(notebook_info)
```

Arguments

notebook_info	The configuration data from the Databricks Notebook environment

Value

A list of tags to be set by the run context when creating MLflow runs in the current Databricks Notebook environment

mlflow_client *Initialize an MLflow Client*

Description

Initializes and returns an MLflow client that communicates with the tracking server or store at the specified URI.

Usage

```
mlflow_client(tracking_uri = NULL)
```

Arguments

tracking_uri	The tracking URI. If not provided, defaults to the service set by ‘mlflow_set_tracking_uri()‘.

`mlflow_create_experiment` *Create Experiment*

Description

Creates an MLflow experiment and returns its id.

Usage

```
mlflow_create_experiment(
  name,
  artifact_location = NULL,
  client = NULL,
  tags = NULL
)
```

Arguments

`name`	The name of the experiment to create.
`artifact_location`	Location where all artifacts for this experiment are stored. If not provided, the remote server will select an appropriate default.
`client`	(Optional) An MLflow client object returned from mlflow_client. If specified, MLflow will use the tracking server associated with the passed-in client. If unspecified (the common case), MLflow will use the tracking server associated with the current tracking URI.
`tags`	Experiment tags to set on the experiment upon experiment creation.

`mlflow_create_model_version` *Create a model version*

Description

Create a model version

Usage

```
mlflow_create_model_version(
  name,
  source,
  run_id = NULL,
  tags = NULL,
```

```
  run_link = NULL,
  description = NULL,
  client = NULL
)
```

Arguments

name	Register model under this name.
source	URI indicating the location of the model artifacts.
run_id	MLflow run ID for correlation, if 'source' was generated by an experiment run in MLflow Tracking.
tags	Additional metadata.
run_link	MLflow run link - This is the exact link of the run that generated this model version.
description	Description for model version.
client	(Optional) An MLflow client object returned from mlflow_client. If specified, MLflow will use the tracking server associated with the passed-in client. If unspecified (the common case), MLflow will use the tracking server associated with the current tracking URI.

---

mlflow_create_registered_model

Create registered model

Description

Creates a new registered model in the model registry

Usage

```
mlflow_create_registered_model(
  name,
  tags = NULL,
  description = NULL,
  client = NULL
)
```

Arguments

name	The name of the model to create.
tags	Additional metadata for the registered model (Optional).
description	Description for the registered model (Optional).
client	(Optional) An MLflow client object returned from mlflow_client. If specified, MLflow will use the tracking server associated with the passed-in client. If unspecified (the common case), MLflow will use the tracking server associated with the current tracking URI.

`mlflow_delete_experiment` *Delete Experiment*

Description

Marks an experiment and associated runs, params, metrics, etc. for deletion. If the experiment uses FileStore, artifacts associated with experiment are also deleted.

Usage

```
mlflow_delete_experiment(experiment_id, client = NULL)
```

Arguments

`experiment_id`	ID of the associated experiment. This field is required.
`client`	(Optional) An MLflow client object returned from mlflow_client. If specified, MLflow will use the tracking server associated with the passed-in client. If unspecified (the common case), MLflow will use the tracking server associated with the current tracking URI.

`mlflow_delete_model_version` *Delete a model version*

Description

Delete a model version

Usage

```
mlflow_delete_model_version(name, version, client = NULL)
```

Arguments

`name`	Name of the registered model.
`version`	Model version number.
`client`	(Optional) An MLflow client object returned from mlflow_client. If specified, MLflow will use the tracking server associated with the passed-in client. If unspecified (the common case), MLflow will use the tracking server associated with the current tracking URI.

mlflow_delete_registered_model *Delete registered model*

Description

Deletes an existing registered model by name

Usage

```
mlflow_delete_registered_model(name, client = NULL)
```

Arguments

name	The name of the model to delete
client	(Optional) An MLflow client object returned from mlflow_client. If specified, MLflow will use the tracking server associated with the passed-in client. If unspecified (the common case), MLflow will use the tracking server associated with the current tracking URI.

mlflow_delete_run *Delete a Run*

Description

Deletes the run with the specified ID.

Usage

```
mlflow_delete_run(run_id, client = NULL)
```

Arguments

run_id	Run ID.
client	(Optional) An MLflow client object returned from mlflow_client. If specified, MLflow will use the tracking server associated with the passed-in client. If unspecified (the common case), MLflow will use the tracking server associated with the current tracking URI.

mlflow_delete_tag *Delete Tag*

Description

Deletes a tag on a run. This is irreversible. Tags are run metadata that can be updated during a run and after a run completes.

Usage

```
mlflow_delete_tag(key, run_id = NULL, client = NULL)
```

Arguments

key	Name of the tag. Maximum size is 255 bytes. This field is required.
run_id	Run ID.
client	(Optional) An MLflow client object returned from mlflow_client. If specified, MLflow will use the tracking server associated with the passed-in client. If unspecified (the common case), MLflow will use the tracking server associated with the current tracking URI.

mlflow_download_artifacts *Download Artifacts*

Description

Download an artifact file or directory from a run to a local directory if applicable, and return a local path for it.

Usage

```
mlflow_download_artifacts(path, run_id = NULL, client = NULL)
```

Arguments

path	Relative source path to the desired artifact.
run_id	Run ID.
client	(Optional) An MLflow client object returned from mlflow_client. If specified, MLflow will use the tracking server associated with the passed-in client. If unspecified (the common case), MLflow will use the tracking server associated with the current tracking URI.

mlflow_end_run *End a Run*

Description

Terminates a run. Attempts to end the current active run if ‘run_id‘ is not specified.

Usage

```
mlflow_end_run(
  status = c("FINISHED", "FAILED", "KILLED"),
  end_time = NULL,
  run_id = NULL,
  client = NULL
)
```

Arguments

status	Updated status of the run. Defaults to ‘FINISHED‘. Can also be set to "FAILED" or "KILLED".
end_time	Unix timestamp of when the run ended in milliseconds.
run_id	Run ID.
client	(Optional) An MLflow client object returned from mlflow_client. If specified, MLflow will use the tracking server associated with the passed-in client. If unspecified (the common case), MLflow will use the tracking server associated with the current tracking URI.

mlflow_get_experiment *Get Experiment*

Description

Gets metadata for an experiment and a list of runs for the experiment. Attempts to obtain the active experiment if both ‘experiment_id‘ and ‘name‘ are unspecified.

Usage

```
mlflow_get_experiment(experiment_id = NULL, name = NULL, client = NULL)
```

Arguments

experiment_id	ID of the experiment.
name	The experiment name. Only one of 'name' or 'experiment_id' should be specified.
client	(Optional) An MLflow client object returned from mlflow_client. If specified, MLflow will use the tracking server associated with the passed-in client. If unspecified (the common case), MLflow will use the tracking server associated with the current tracking URI.

---

mlflow_get_latest_versions

Get latest model versions

Description

Retrieves a list of the latest model versions for a given model.

Usage

```
mlflow_get_latest_versions(name, stages = list(), client = NULL)
```

Arguments

name	Name of the model.
stages	A list of desired stages. If the input list is NULL, return latest versions for ALL_STAGES.
client	(Optional) An MLflow client object returned from mlflow_client. If specified, MLflow will use the tracking server associated with the passed-in client. If unspecified (the common case), MLflow will use the tracking server associated with the current tracking URI.

---

mlflow_get_metric_history

Get Metric History

Description

Get a list of all values for the specified metric for a given run.

Usage

```
mlflow_get_metric_history(metric_key, run_id = NULL, client = NULL)
```

Arguments

metric_key	Name of the metric.
run_id	Run ID.
client	(Optional) An MLflow client object returned from mlflow_client. If specified, MLflow will use the tracking server associated with the passed-in client. If unspecified (the common case), MLflow will use the tracking server associated with the current tracking URI.

---

mlflow_get_model_version

Get a model version

Description

Get a model version

Usage

```
mlflow_get_model_version(name, version, client = NULL)
```

Arguments

name	Name of the registered model.
version	Model version number.
client	(Optional) An MLflow client object returned from mlflow_client. If specified, MLflow will use the tracking server associated with the passed-in client. If unspecified (the common case), MLflow will use the tracking server associated with the current tracking URI.

---

mlflow_get_registered_model

Get a registered model

Description

Retrieves a registered model from the Model Registry.

Usage

```
mlflow_get_registered_model(name, client = NULL)
```

Arguments

name	The name of the model to retrieve.
client	(Optional) An MLflow client object returned from mlflow_client. If specified, MLflow will use the tracking server associated with the passed-in client. If unspecified (the common case), MLflow will use the tracking server associated with the current tracking URI.

---

mlflow_get_run *Get Run*

Description

Gets metadata, params, tags, and metrics for a run. Returns a single value for each metric key: the most recently logged metric value at the largest step.

Usage

```
mlflow_get_run(run_id = NULL, client = NULL)
```

Arguments

run_id	Run ID.
client	(Optional) An MLflow client object returned from mlflow_client. If specified, MLflow will use the tracking server associated with the passed-in client. If unspecified (the common case), MLflow will use the tracking server associated with the current tracking URI.

---

mlflow_get_tracking_uri *Get Remote Tracking URI*

Description

Gets the remote tracking URI.

Usage

```
mlflow_get_tracking_uri()
```

mlflow_id *Get Run or Experiment ID*

Description

Extracts the ID of the run or experiment.

Usage

```
mlflow_id(object)

## S3 method for class 'mlflow_run'
mlflow_id(object)

## S3 method for class 'mlflow_experiment'
mlflow_id(object)
```

Arguments

object	An ‘mlflow_run‘ or ‘mlflow_experiment‘ object.

mlflow_list_artifacts *List Artifacts*

Description

Gets a list of artifacts.

Usage

```
mlflow_list_artifacts(path = NULL, run_id = NULL, client = NULL)
```

Arguments

path	The run’s relative artifact path to list from. If not specified, it is set to the root artifact path
run_id	Run ID.
client	(Optional) An MLflow client object returned from mlflow_client. If specified, MLflow will use the tracking server associated with the passed-in client. If unspecified (the common case), MLflow will use the tracking server associated with the current tracking URI.

mlflow_load_flavor *Load MLflow Model Flavor*

Description

Loads an MLflow model using a specific flavor. This method is called internally by mlflow_load_model, but is exposed for package authors to extend the supported MLflow models. See https://mlflow.org/docs/latest/models.html#st format for more info on MLflow model flavors.

Usage

```
mlflow_load_flavor(flavor, model_path)
```

Arguments

flavor	An MLflow flavor object loaded by mlflow_load_model, with class loaded from the flavor field in an MLmodel file.
model_path	The path to the MLflow model wrapped in the correct class.

mlflow_load_model *Load MLflow Model*

Description

Loads an MLflow model. MLflow models can have multiple model flavors. Not all flavors / models can be loaded in R. This method by default searches for a flavor supported by R/MLflow.

Usage

```
mlflow_load_model(model_uri, flavor = NULL, client = mlflow_client())
```

Arguments

model_uri	The location, in URI format, of the MLflow model.
flavor	Optional flavor specification (string). Can be used to load a particular flavor in case there are multiple flavors available.
client	(Optional) An MLflow client object returned from mlflow_client. If specified, MLflow will use the tracking server associated with the passed-in client. If unspecified (the common case), MLflow will use the tracking server associated with the current tracking URI.

Details

The URI scheme must be supported by MLflow - i.e. there has to be an MLflow artifact repository corresponding to the scheme of the URI. The content is expected to point to a directory containing MLmodel. The following are examples of valid model uris:

- “file:///absolute/path/to/local/model“ - “file:relative/path/to/local/model“ - “s3://my_bucket/path/to/model“
- “runs:/<mlflow_run_id>/run-relative/path/to/model“ - “models:/<model_name>/<model_version>“
- “models:/<model_name>/<stage>“

For more information about supported URI schemes, see the Artifacts Documentation at https://www.mlflow.org/docs/latest/tr
stores.

`mlflow_log_artifact` *Log Artifact*

Description

Logs a specific file or directory as an artifact for a run.

Usage

```
mlflow_log_artifact(path, artifact_path = NULL, run_id = NULL, client = NULL)
```

Arguments

`path`	The file or directory to log as an artifact.
`artifact_path`	Destination path within the run’s artifact URI.
`run_id`	Run ID.
`client`	(Optional) An MLflow client object returned from mlflow_client. If specified, MLflow will use the tracking server associated with the passed-in client. If unspecified (the common case), MLflow will use the tracking server associated with the current tracking URI.

Details

When logging to Amazon S3, ensure that you have the s3:PutObject, s3:GetObject, s3:ListBucket, and s3:GetBucketLocation permissions on your bucket.

Additionally, at least the `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY` environment variables must be set to the corresponding key and secrets provided by Amazon IAM.

`mlflow_log_batch` *Log Batch*

Description

Log a batch of metrics, params, and/or tags for a run. The server will respond with an error (non-200 status code) if any data failed to be persisted. In case of error (due to internal server error or an invalid request), partial data may be written.

Usage

```
mlflow_log_batch(
  metrics = NULL,
  params = NULL,
  tags = NULL,
  run_id = NULL,
  client = NULL
)
```

Arguments

`metrics`	A dataframe of metrics to log, containing the following columns: "key", "value", "step", "timestamp". This dataframe cannot contain any missing ('NA') entries.
`params`	A dataframe of params to log, containing the following columns: "key", "value". This dataframe cannot contain any missing ('NA') entries.
`tags`	A dataframe of tags to log, containing the following columns: "key", "value". This dataframe cannot contain any missing ('NA') entries.
`run_id`	Run ID.
`client`	(Optional) An MLflow client object returned from mlflow_client. If specified, MLflow will use the tracking server associated with the passed-in client. If unspecified (the common case), MLflow will use the tracking server associated with the current tracking URI.

`mlflow_log_metric` *Log Metric*

Description

Logs a metric for a run. Metrics key-value pair that records a single float measure. During a single execution of a run, a particular metric can be logged several times. The MLflow Backend keeps track of historical metric values along two axes: timestamp and step.

Usage

```
mlflow_log_metric(
  key,
  value,
  timestamp = NULL,
  step = NULL,
  run_id = NULL,
  client = NULL
)
```

Arguments

key	Name of the metric.
value	Float value for the metric being logged.
timestamp	Timestamp at which to log the metric. Timestamp is rounded to the nearest integer. If unspecified, the number of milliseconds since the Unix epoch is used.
step	Step at which to log the metric. Step is rounded to the nearest integer. If unspecified, the default value of zero is used.
run_id	Run ID.
client	(Optional) An MLflow client object returned from mlflow_client. If specified, MLflow will use the tracking server associated with the passed-in client. If unspecified (the common case), MLflow will use the tracking server associated with the current tracking URI.

---

mlflow_log_model *Log Model*

Description

Logs a model for this run. Similar to 'mlflow_save_model()' but stores model as an artifact within the active run.

Usage

```
mlflow_log_model(model, artifact_path, ...)
```

Arguments

model	The model that will perform a prediction.
artifact_path	Destination path where this MLflow compatible model will be saved.
...	Optional additional arguments passed to 'mlflow_save_model()' when persisting the model. For example, 'conda_env = /path/to/conda.yaml' may be passed to specify a conda dependencies file for flavors (e.g. keras) that support conda environments.

mlflow_log_param *Log Parameter*

Description

Logs a parameter for a run. Examples are params and hyperparams used for ML training, or constant dates and values used in an ETL pipeline. A param is a STRING key-value pair. For a run, a single parameter is allowed to be logged only once.

Usage

```
mlflow_log_param(key, value, run_id = NULL, client = NULL)
```

Arguments

key	Name of the parameter.
value	String value of the parameter.
run_id	Run ID.
client	(Optional) An MLflow client object returned from mlflow_client. If specified, MLflow will use the tracking server associated with the passed-in client. If unspecified (the common case), MLflow will use the tracking server associated with the current tracking URI.

mlflow_param *Read Command-Line Parameter*

Description

Reads a command-line parameter passed to an MLflow project MLflow allows you to define named, typed input parameters to your R scripts via the mlflow_param API. This is useful for experimentation, e.g. tracking multiple invocations of the same script with different parameters.

Usage

```
mlflow_param(name, default = NULL, type = NULL, description = NULL)
```

Arguments

name	The name of the parameter.
default	The default value of the parameter.
type	Type of this parameter. Required if ‘default‘ is not set. If specified, must be one of "numeric", "integer", or "string".
description	Optional description for the parameter.

Examples

```
## Not run:
# This parametrized script trains a GBM model on the Iris dataset and can be run as an MLflow
# project. You can run this script (assuming it's saved at /some/directory/params_example.R)
# with custom parameters via:
# mlflow_run(entry_point = "params_example.R", uri = "/some/directory",
#   parameters = list(num_trees = 200, learning_rate = 0.1))
install.packages("gbm")
library(mlflow)
library(gbm)
# define and read input parameters
num_trees <- mlflow_param(name = "num_trees", default = 200, type = "integer")
lr <- mlflow_param(name = "learning_rate", default = 0.1, type = "numeric")
# use params to fit a model
ir.adaboost <- gbm(Species ~., data=iris, n.trees=num_trees, shrinkage=lr)

## End(Not run)
```

`mlflow_predict` *Generate Prediction with MLflow Model*

Description

Performs prediction over a model loaded using `mlflow_load_model()`, to be used by package authors to extend the supported MLflow models.

Usage

```
mlflow_predict(model, data, ...)
```

Arguments

`model`	The loaded MLflow model flavor.
`data`	A data frame to perform scoring.
`...`	Optional additional arguments passed to underlying predict methods.

`mlflow_register_external_observer`
Register an external MLflow observer

Description

Registers an external MLflow observer that will receive a ‘register_tracking_event(event_name, data)‘ callback on any model tracking event such as "create_run", "delete_run", or "log_metric". Each observer should have a ‘register_tracking_event(event_name, data)‘ callback accepting a character vector ‘event_name‘ specifying the name of the tracking event, and ‘data‘ containing a list of attributes of the event. The callback should be non-blocking, and ideally should complete instantaneously. Any exception thrown from the callback will be ignored.

Usage

```
mlflow_register_external_observer(observer)
```

Arguments

observer	The observer object (see example)

Examples

```
library(mlflow)

observer <- structure(list())
observer$register_tracking_event <- function(event_name, data) {
  print(event_name)
  print(data)
}
mlflow_register_external_observer(observer)
```

---

mlflow_rename_experiment

Rename Experiment

Description

Renames an experiment.

Usage

```
mlflow_rename_experiment(new_name, experiment_id = NULL, client = NULL)
```

Arguments

new_name	The experiment’s name will be changed to this. The new name must be unique.
experiment_id	ID of the associated experiment. This field is required.
client	(Optional) An MLflow client object returned from mlflow_client. If specified, MLflow will use the tracking server associated with the passed-in client. If unspecified (the common case), MLflow will use the tracking server associated with the current tracking URI.

mlflow_rename_registered_model *Rename a registered model*

Description

Renames a model in the Model Registry.

Usage

```
mlflow_rename_registered_model(name, new_name, client = NULL)
```

Arguments

name	The current name of the model.
new_name	The new name for the model.
client	(Optional) An MLflow client object returned from mlflow_client. If specified, MLflow will use the tracking server associated with the passed-in client. If unspecified (the common case), MLflow will use the tracking server associated with the current tracking URI.

mlflow_restore_experiment *Restore Experiment*

Description

Restores an experiment marked for deletion. This also restores associated metadata, runs, metrics, and params. If experiment uses FileStore, underlying artifacts associated with experiment are also restored.

Usage

```
mlflow_restore_experiment(experiment_id, client = NULL)
```

Arguments

experiment_id	ID of the associated experiment. This field is required.
client	(Optional) An MLflow client object returned from mlflow_client. If specified, MLflow will use the tracking server associated with the passed-in client. If unspecified (the common case), MLflow will use the tracking server associated with the current tracking URI.

Details

Throws 'RESOURCE_DOES_NOT_EXIST' if the experiment was never created or was permanently deleted.

mlflow_restore_run *Restore a Run*

Description

Restores the run with the specified ID.

Usage

```
mlflow_restore_run(run_id, client = NULL)
```

Arguments

run_id	Run ID.
client	(Optional) An MLflow client object returned from mlflow_client. If specified, MLflow will use the tracking server associated with the passed-in client. If unspecified (the common case), MLflow will use the tracking server associated with the current tracking URI.

mlflow_rfunc_serve *Serve an RFunc MLflow Model*

Description

Serves an RFunc MLflow model as a local REST API server. This interface provides similar functionality to "mlflow models serve" cli command, however, it can only be used to deploy models that include RFunc flavor. The deployed server supports standard mlflow models interface with /ping and /invocation endpoints. In addition, R function models also support deprecated /predict endpoint for generating predictions. The /predict endpoint will be removed in a future version of mlflow.

Usage

```
mlflow_rfunc_serve(
  model_uri,
  host = "127.0.0.1",
  port = 8090,
  daemonized = FALSE,
  browse = !daemonized,
  ...
)
```

Arguments

model_uri	The location, in URI format, of the MLflow model.
host	Address to use to serve model, as a string.
port	Port to use to serve model, as numeric.
daemonized	Makes ‘httpuv‘ server daemonized so R interactive sessions are not blocked to handle requests. To terminate a daemonized server, call ‘httpuv::stopDaemonizedServer()‘ with the handle returned from this call.
browse	Launch browser with serving landing page?
...	Optional arguments passed to ‘mlflow_predict()‘.

Details

The URI scheme must be supported by MLflow - i.e. there has to be an MLflow artifact repository corresponding to the scheme of the URI. The content is expected to point to a directory containing MLmodel. The following are examples of valid model uris:

- “file:///absolute/path/to/local/model“ - “file:relative/path/to/local/model“ - “s3://my_bucket/path/to/model“
- “runs:/<mlflow_run_id>/run-relative/path/to/model“ - “models:/<model_name>/<model_version>“
- “models:/<model_name>/<stage>“

For more information about supported URI schemes, see the Artifacts Documentation at https://www.mlflow.org/docs/latest/tr
stores.

Examples

```
## Not run:
library(mlflow)

# save simple model with constant prediction
mlflow_save_model(function(df) 1, "mlflow_constant")

# serve an existing model over a web interface
mlflow_rfunc_serve("mlflow_constant")

# request prediction from server
httr::POST("http://127.0.0.1:8090/predict/")

## End(Not run)
```

---

mlflow_run *Run an MLflow Project*

Description

Wrapper for the ‘mlflow run‘ CLI command. See https://www.mlflow.org/docs/latest/cli.html#mlflow-run for more info.

Usage

```
mlflow_run(
  uri = ".",
  entry_point = NULL,
  version = NULL,
  parameters = NULL,
  experiment_id = NULL,
  experiment_name = NULL,
  backend = NULL,
  backend_config = NULL,
  env_manager = NULL,
  storage_dir = NULL
)
```

Arguments

uri	A directory containing modeling scripts, defaults to the current directory.
entry_point	Entry point within project, defaults to ‘main‘ if not specified.
version	Version of the project to run, as a Git commit reference for Git projects.
parameters	A list of parameters.
experiment_id	ID of the experiment under which to launch the run.
experiment_name	Name of the experiment under which to launch the run.
backend	Execution backend to use for run.
backend_config	Path to JSON file which will be passed to the backend. For the Databricks backend, it should describe the cluster to use when launching a run on Databricks.
env_manager	If specified, create an environment for the project using the specified environment manager. Available options are ’local’, ’virtualenv’, and ’conda’.
storage_dir	Valid only when ‘backend‘ is local. MLflow downloads artifacts from distributed URIs passed to parameters of type ‘path‘ to subdirectories of ‘storage_dir‘.

Value

The run associated with this run.

Examples

```
## Not run:
# This parametrized script trains a GBM model on the Iris dataset and can be run as an MLflow
# project. You can run this script (assuming it's saved at /some/directory/params_example.R)
# with custom parameters via:
# mlflow_run(entry_point = "params_example.R", uri = "/some/directory",
#   parameters = list(num_trees = 200, learning_rate = 0.1))
install.packages("gbm")
library(mlflow)
library(gbm)
```

```
# define and read input parameters
num_trees <- mlflow_param(name = "num_trees", default = 200, type = "integer")
lr <- mlflow_param(name = "learning_rate", default = 0.1, type = "numeric")
# use params to fit a model
ir.adaboost <- gbm(Species ~., data=iris, n.trees=num_trees, shrinkage=lr)

## End(Not run)
```

`mlflow_save_model.crate` *Save Model for MLflow*

Description

Saves model in MLflow format that can later be used for prediction and serving. This method is generic to allow package authors to save custom model types.

Usage

```
## S3 method for class 'crate'
mlflow_save_model(model, path, model_spec = list(), ...)

mlflow_save_model(model, path, model_spec = list(), ...)

## S3 method for class 'H2OModel'
mlflow_save_model(model, path, model_spec = list(), conda_env = NULL, ...)

## S3 method for class 'keras.engine.training.Model'
mlflow_save_model(model, path, model_spec = list(), conda_env = NULL, ...)

## S3 method for class 'xgb.Booster'
mlflow_save_model(model, path, model_spec = list(), conda_env = NULL, ...)
```

Arguments

`model`	The model that will perform a prediction.
`path`	Destination path where this MLflow compatible model will be saved.
`model_spec`	MLflow model config this model flavor is being added to.
`...`	Optional additional arguments.
`conda_env`	Path to Conda dependencies file.

`mlflow_search_experiments` *Search Experiments*

Description

Search for experiments that satisfy specified criteria.

Usage

```
mlflow_search_experiments(
  filter = NULL,
  experiment_view_type = c("ACTIVE_ONLY", "DELETED_ONLY", "ALL"),
  max_results = 1000,
  order_by = list(),
  page_token = NULL,
  client = NULL
)
```

Arguments

`filter`	A filter expression used to identify specific experiments. The syntax is a subset of SQL which allows only ANDing together binary operations. Examples: "attribute.name = 'MyExperiment'", "tags.problem_type = 'iris_regression'"
`experiment_view_type`	Experiment view type. Only experiments matching this view type are returned.
`max_results`	Maximum number of experiments to retrieve.
`order_by`	List of properties to order by. Example: "attribute.name".
`page_token`	Pagination token to go to the next page based on a previous query.
`client`	(Optional) An MLflow client object returned from mlflow_client. If specified, MLflow will use the tracking server associated with the passed-in client. If unspecified (the common case), MLflow will use the tracking server associated with the current tracking URI.

`mlflow_search_registered_models` *List registered models*

Description

Retrieves a list of registered models.

Usage

```
mlflow_search_registered_models(
  filter = NULL,
  max_results = 100,
  order_by = list(),
  page_token = NULL,
  client = NULL
)
```

Arguments

filter	A filter expression used to identify specific registered models. The syntax is a subset of SQL which allows only ANDing together binary operations. Example: "name = 'my_model_name' and tag.key = 'value1'"
max_results	Maximum number of registered models to retrieve.
order_by	List of registered model properties to order by. Example: "name".
page_token	Pagination token to go to the next page based on a previous query.
client	(Optional) An MLflow client object returned from mlflow_client. If specified, MLflow will use the tracking server associated with the passed-in client. If unspecified (the common case), MLflow will use the tracking server associated with the current tracking URI.

---

mlflow_search_runs *Search Runs*

Description

Search for runs that satisfy expressions. Search expressions can use Metric and Param keys.

Usage

```
mlflow_search_runs(
  filter = NULL,
  run_view_type = c("ACTIVE_ONLY", "DELETED_ONLY", "ALL"),
  experiment_ids = NULL,
  order_by = list(),
  client = NULL
)
```

Arguments

filter	A filter expression over params, metrics, and tags, allowing returning a subset of runs. The syntax is a subset of SQL which allows only ANDing together binary operations between a param/metric/tag and a constant.
run_view_type	Run view type.

experiment_ids	List of string experiment IDs (or a single string experiment ID) to search over. Attempts to use active experiment if not specified.
order_by	List of properties to order by. Example: "metrics.acc DESC".
client	(Optional) An MLflow client object returned from mlflow_client. If specified, MLflow will use the tracking server associated with the passed-in client. If unspecified (the common case), MLflow will use the tracking server associated with the current tracking URI.

`mlflow_server` *Run MLflow Tracking Server*

Description

Wrapper for 'mlflow server'.

Usage

```
mlflow_server(
  file_store = "mlruns",
  default_artifact_root = NULL,
  host = "127.0.0.1",
  port = 5000,
  workers = NULL,
  static_prefix = NULL,
  serve_artifacts = FALSE
)
```

Arguments

file_store	The root of the backing file store for experiment and run data.
default_artifact_root	Local or S3 URI to store artifacts in, for newly created experiments.
host	The network address to listen on (default: 127.0.0.1).
port	The port to listen on (default: 5000).
workers	Number of gunicorn worker processes to handle requests (default: 4).
static_prefix	A prefix which will be prepended to the path of all static paths.
serve_artifacts	A flag specifying whether or not to enable artifact serving (default: FALSE).

`mlflow_set_experiment` *Set Experiment*

Description

Sets an experiment as the active experiment. Either the name or ID of the experiment can be provided. If the a name is provided but the experiment does not exist, this function creates an experiment with provided name. Returns the ID of the active experiment.

Usage

```
mlflow_set_experiment(
  experiment_name = NULL,
  experiment_id = NULL,
  artifact_location = NULL
)
```

Arguments

`experiment_name`	Name of experiment to be activated.
`experiment_id`	ID of experiment to be activated.
`artifact_location`	Location where all artifacts for this experiment are stored. If not provided, the remote server will select an appropriate default.

`mlflow_set_experiment_tag` *Set Experiment Tag*

Description

Sets a tag on an experiment with the specified ID. Tags are experiment metadata that can be updated.

Usage

```
mlflow_set_experiment_tag(key, value, experiment_id = NULL, client = NULL)
```

Arguments

`key`	Name of the tag. All storage backends are guaranteed to support key values up to 250 bytes in size. This field is required.
`value`	String value of the tag being logged. All storage backends are guaranteed to support key values up to 5000 bytes in size. This field is required.

experiment_id	ID of the experiment.
client	(Optional) An MLflow client object returned from mlflow_client. If specified, MLflow will use the tracking server associated with the passed-in client. If unspecified (the common case), MLflow will use the tracking server associated with the current tracking URI.

`mlflow_set_model_version_tag` *Set Model version tag*

Description

Set a tag for the model version. When stage is set, tag will be set for latest model version of the stage. Setting both version and stage parameter will result in error.

Usage

```
mlflow_set_model_version_tag(
  name,
  version = NULL,
  key = NULL,
  value = NULL,
  stage = NULL,
  client = NULL
)
```

Arguments

name	Registered model name.
version	Registered model version.
key	Tag key to log. key is required.
value	Tag value to log. value is required.
stage	Registered model stage.
client	(Optional) An MLflow client object returned from mlflow_client. If specified, MLflow will use the tracking server associated with the passed-in client. If unspecified (the common case), MLflow will use the tracking server associated with the current tracking URI.

mlflow_set_tag *Set Tag*

Description

Sets a tag on a run. Tags are run metadata that can be updated during a run and after a run completes.

Usage

```
mlflow_set_tag(key, value, run_id = NULL, client = NULL)
```

Arguments

key	Name of the tag. Maximum size is 255 bytes. This field is required.
value	String value of the tag being logged. Maximum size is 500 bytes. This field is required.
run_id	Run ID.
client	(Optional) An MLflow client object returned from mlflow_client. If specified, MLflow will use the tracking server associated with the passed-in client. If unspecified (the common case), MLflow will use the tracking server associated with the current tracking URI.

mlflow_set_tracking_uri *Set Remote Tracking URI*

Description

Specifies the URI to the remote MLflow server that will be used to track experiments.

Usage

```
mlflow_set_tracking_uri(uri)
```

Arguments

uri	The URI to the remote MLflow server.

`mlflow_start_run` *Start Run*

Description

Starts a new run. If ‘client‘ is not provided, this function infers contextual information such as source name and version, and also registers the created run as the active run. If ‘client‘ is provided, no inference is done, and additional arguments such as ‘start_time‘ can be provided.

Usage

```
mlflow_start_run(
  run_id = NULL,
  experiment_id = NULL,
  start_time = NULL,
  tags = NULL,
  client = NULL,
  nested = FALSE
)
```

Arguments

`run_id`	If specified, get the run with the specified UUID and log metrics and params under that run. The run’s end time is unset and its status is set to running, but the run’s other attributes remain unchanged.
`experiment_id`	Used only when ‘run_id‘ is unspecified. ID of the experiment under which to create the current run. If unspecified, the run is created under a new experiment with a randomly generated name.
`start_time`	Unix timestamp of when the run started in milliseconds. Only used when ‘client‘ is specified.
`tags`	Additional metadata for run in key-value pairs. Only used when ‘client‘ is specified.
`client`	(Optional) An MLflow client object returned from mlflow_client. If specified, MLflow will use the tracking server associated with the passed-in client. If unspecified (the common case), MLflow will use the tracking server associated with the current tracking URI.
`nested`	Controls whether the run to be started is nested in a parent run. ‘TRUE‘ creates a nest run.

Examples

```
## Not run:
with(mlflow_start_run(), {
  mlflow_log_metric("test", 10)
})
```

```
## End(Not run)
```

`mlflow_transition_model_version_stage` *Transition ModelVersion Stage*

Description

Transition a model version to a different stage.

Usage

```
mlflow_transition_model_version_stage(
  name,
  version,
  stage,
  archive_existing_versions = FALSE,
  client = NULL
)
```

Arguments

`name`	Name of the registered model.
`version`	Model version number.
`stage`	Transition ‘model_version‘ to this stage.
`archive_existing_versions`	(Optional)
`client`	(Optional) An MLflow client object returned from mlflow_client. If specified, MLflow will use the tracking server associated with the passed-in client. If unspecified (the common case), MLflow will use the tracking server associated with the current tracking URI.

`mlflow_ui` *Run MLflow User Interface*

Description

Launches the MLflow user interface.

Usage

```
mlflow_ui(client, ...)
```

Arguments

client	(Optional) An MLflow client object returned from mlflow_client. If specified, MLflow will use the tracking server associated with the passed-in client. If unspecified (the common case), MLflow will use the tracking server associated with the current tracking URI.
...	Optional arguments passed to ‘mlflow_server()‘ when ‘x‘ is a path to a file store.

Examples

```
## Not run:
library(mlflow)

# launch mlflow ui locally
mlflow_ui()

# launch mlflow ui for existing mlflow server
mlflow_set_tracking_uri("http://tracking-server:5000")
mlflow_ui()

## End(Not run)
```

---

mlflow_update_model_version

Update model version

Description

Updates a model version

Usage

```
mlflow_update_model_version(name, version, description, client = NULL)
```

Arguments

name	Name of the registered model.
version	Model version number.
description	Description of this model version.
client	(Optional) An MLflow client object returned from mlflow_client. If specified, MLflow will use the tracking server associated with the passed-in client. If unspecified (the common case), MLflow will use the tracking server associated with the current tracking URI.

`mlflow_update_registered_model` *Update a registered model*

Description

Updates a model in the Model Registry.

Usage

```
mlflow_update_registered_model(name, description, client = NULL)
```

Arguments

`name`	The name of the registered model.
`description`	The updated description for this registered model.
`client`	(Optional) An MLflow client object returned from mlflow_client. If specified, MLflow will use the tracking server associated with the passed-in client. If unspecified (the common case), MLflow will use the tracking server associated with the current tracking URI.

Index