Package 'mlflow'

May 9, 2021

Type Package

Title Interface to 'MLflow'

Version 1.17.0

Maintainer Matei Zaharia <matei@databricks.com>

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, forge, fs, git2r, glue, httpuv, httr, ini, jsonlite, openssl, processx, purrr, reticulate, rlang (>= 0.2.0), swagger, tibble (>= 2.0.0), withr, xml2, yaml, zeallot

Suggests carrier, covr, h2o, keras, lintr, mleap, sparklyr, stringi (< 1.4.4), testthat (>= 2.0.0), xgboost

Encoding UTF-8

RoxygenNote 7.1.1

Collate 'cli.R' 'databricks-utils.R' 'globals.R' 'imports.R' 'install.R' 'logging.R' 'mlflow-package.R' 'model-crate.R' 'model-python.R' 'model.R' 'model-utils.R' 'model-h2o.R' 'model-keras.R' 'model-mleap.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 Matei Zaharia [aut, cre],
Javier Luraschi [aut],
Kevin Kuo [aut] (<https://orcid.org/0000-0001-7803-7901>),
RStudio [cph]

Repository CRAN

Date/Publication 2021-05-08 23:40:02 UTC

R topics documented:

`install_mlflow` *Install MLflow*

Description

Installs auxiliary dependencies of MLflow (e.g. the MLflow CLI). As a one-time setup step, you must run install_mlflow() to install these dependencies before calling other MLflow APIs.

Usage

```
install_mlflow(python_version = "3.6")
```

Arguments

`python_version`	Optional Python version to use within conda environment created for installing the MLflow CLI. If unspecified, defaults to using Python 3.6

Details

install_mlflow() requires Python and Conda to be installed. See https://www.python.org/getit/ and https://docs.conda.io/projects/conda/en/latest/user-guide/install/.

Alternatively, you can set MLFLOW_PYTHON_BIN and MLFLOW_BIN environment variables instead. MLFLOW_PYTHON_BIN should point to python executable and MLFLOW_BIN to mlflow cli executable. These variables allow you to use custom mlflow installation. Note that there may be some compatibility issues if the custom mlflow version does not match the version of the R package.

Examples

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

## End(Not run)
```

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)
```

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.

`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_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_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_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_list_experiments *List Experiments*

Description

Gets a list of all experiments.

Usage

```
mlflow_list_experiments(
  view_type = c("ACTIVE_ONLY", "DELETED_ONLY", "ALL"),
  client = NULL
)
```

Arguments

view_type	Qualifier for type of experiments to be returned. Defaults to 'ACTIVE_ONLY'.
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_list_run_infos *List Run Infos*

Description

Returns a tibble whose columns contain run metadata (run ID, etc) for all runs under the specified experiment.

Usage

```
mlflow_list_run_infos(
  run_view_type = c("ACTIVE_ONLY", "DELETED_ONLY", "ALL"),
  experiment_id = NULL,
  client = NULL
)
```

Arguments

run_view_type	Run view type.
experiment_id	Experiment ID. Attempts to use the active experiment if not 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_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_maybe_create_conda_env *Create conda env used by MLflow if it doesn't already exist*

Description

Create conda env used by MLflow if it doesn't already exist

Usage

```
mlflow_maybe_create_conda_env(python_version)
```

Arguments

python_version	Python version to use within conda environment created for installing the MLflow CLI.

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_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,
  no_conda = FALSE,
  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.
no_conda	If specified, assume that MLflow is running within a Conda environment with the necessary dependencies for the current project instead of attempting to create a new Conda environment. Only valid if running locally.
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 'ml_pipeline_model'
mlflow_save_model(
  model,
  path,
  model_spec = list(),
  conda_env = NULL,
  sample_input = 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.
sample_input	Sample Spark DataFrame input that the model can evaluate. This is required by MLeap for data schema inference.

---

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 = 4,
  static_prefix = NULL
)
```

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.

---

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_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_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)
install_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)
```

`uninstall_mlflow` *Uninstall MLflow*

Description

Uninstalls MLflow by removing the Conda environment.

Usage

```
uninstall_mlflow()
```

Examples

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

## End(Not run)
```

Index