Package 'mlflow'

November 12, 2018

Type Package

Title Interface to 'MLflow'

Version 0.8.0

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

SystemRequirements MLflow (https://www.mlflow.org/)

Encoding UTF-8

LazyData true

Depends R (>= 3.1.2)

Imports aws.s3, forge, fs, git2r, httpuv, httr, jsonlite, openssl, processx, reticulate, packrat, purrr, swagger, withr, xml2, yaml, pryr, rlang (>= 0.2.0)

RoxygenNote 6.1.0

Suggests covr, keras, lintr, testthat

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 2018-11-12 22:00:03 UTC

R topics documented:

`crate` *Crate a function to share with another process*

Description

'crate()' creates functions in a self-contained environment (technically, a child of the base environment). This has two advantages:

* They can easily be executed in another process.

* Their effects are reproducible. You can run them locally with the same results as on a different process.

Creating self-contained functions requires some care, see section below.

Usage

```
crate(.fn, ...)
```

Arguments

`.fn`	A fresh formula or function. "Fresh" here means that they should be declared in the call to ‘crate()‘. See examples if you need to crate a function that is already defined. Formulas are converted to purrr-like lambda functions using [rlang::as_function()].
`...`	Arguments to declare in the environment of ‘.fn‘. If a name is supplied, the object is assigned to that name. Otherwise the argument is automatically named after itself.

Creating self-contained functions

* They should call package functions with an explicit ‘::‘ namespace. This includes packages in the default search path with the exception of the base package. For instance ‘var()‘ from the stats package must be called with its namespace prefix: ‘stats::var(x)‘.

* They should declare any data they depend on. You can declare data by supplying additional arguments or by unquoting objects with ‘!¡.

Examples

```
# You can create functions using the ordinary notation:
crate(function(x) stats::var(x))

# Or the formula notation:
crate(~stats::var(.x))

# Declare data by supplying named arguments. You can test you have
# declared all necessary data by calling your crated function:
na_rm <- TRUE
fn <- crate(~stats::var(.x, na.rm = na_rm))
try(fn(1:10))

# Arguments are automatically named after themselves so that the
# following are equivalent:
crate(~stats::var(.x, na.rm = na_rm), na_rm = na_rm)
crate(~stats::var(.x, na.rm = na_rm), na_rm)

# However if you supply a complex expression, do supply a name!
crate(~stats::var(.x, na.rm = na_rm), !na_rm)
crate(~stats::var(.x, na.rm = na_rm), na_rm = na_rm)

# For small data it is handy to unquote instead. Unquoting inlines
# objects inside the function. This is less verbose if your
# function depends on many small objects:
fn <- crate(~stats::var(.x, na.rm = !!na_rm))
fn(1:10)
```

```
# One downside is that the individual sizes of unquoted objects
# won't be shown in the crate printout:
fn


# The function or formula you pass to crate() should defined inside
# the crate() call, i.e. you can't pass an already defined
# function:
fn <- function(x) toupper(x)
try(crate(fn))

# If you really need to crate an existing function, you can
# explicitly set its environment to the crate environment with the
# set_env() function from rlang:
crate(rlang::set_env(fn))
```

`is_crate` *Is an object a crate?*

Description

Is an object a crate?

Usage

```
is_crate(x)
```

Arguments

`x`	An object to test.

`mlflow_active_run` *Active Run*

Description

Retrieves the active run.

Usage

```
mlflow_active_run()
```

mlflow_cli *MLflow Command*

Description

Executes a generic MLflow command through the commmand line interface.

Usage

```
mlflow_cli(..., background = FALSE, echo = TRUE,
  stderr_callback = NULL)
```

Arguments

...	The parameters to pass to the command line.
background	Should this command be triggered as a background task? Defaults to FALSE.
echo	Print the standard output and error to the screen? Defaults to TRUE, does not apply to background tasks.
stderr_callback	NULL, or a function to call for every chunk of the standard error.

Value

A processx task.

Examples

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

mlflow_cli("server", "--help")

## End(Not run)
```

mlflow_create_experiment *Create Experiment*

Description

Creates an MLflow experiment.

Usage

```
mlflow_create_experiment(name, artifact_location = 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.

Details

The fluent API family of functions operate with an implied MLflow client determined by the service set by ‘mlflow_set_tracking_uri()‘. For operations involving a run it adopts the current active run, or, if one does not exist, starts one through the implied service.

---

mlflow_end_run *End a Run*

Description

End an active MLflow run (if there is one).

Usage

```
mlflow_end_run(status = c("FINISHED", "SCHEDULED", "FAILED", "KILLED"))
```

Arguments

status	Updated status of the run. Defaults to ‘FINISHED‘.

Details

The fluent API family of functions operate with an implied MLflow client determined by the service set by ‘mlflow_set_tracking_uri()‘. For operations involving a run it adopts the current active run, or, if one does not exist, starts one through the implied service.

---

mlflow_get_tracking_uri *Get Remote Tracking URI*

Description

Get Remote Tracking URI

Usage

```
mlflow_get_tracking_uri()
```

mlflow_install *Install MLflow*

Description

Installs MLflow for individual use.

Usage

```
mlflow_install()
```

Details

Notice that MLflow requires Python and Conda to be installed, see https://www.python.org/getit/ and https://conda.io/docs/installation.html.

Examples

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

## End(Not run)
```

mlflow_load_flavor *Load MLflow Model Flavor*

Description

Loads an MLflow model flavor, to be used by package authors to extend the supported MLflow models.

Usage

```
mlflow_load_flavor(model_path)
```

Arguments

model_path	The path to the MLflow model wrapped in the correct class.

mlflow_load_model *Load MLflow Model.*

Description

MLflow models can have multiple model flavors. Not all flavors / models can be loaded in R. This method will by default search for a flavor supported by R/mlflow.

Usage

```
mlflow_load_model(model_path, flavor = NULL, run_id = NULL)
```

Arguments

model_path	"Path to the MLflow model. The path is relative to the run with the given run-id or local filesystem path without run-id.
flavor	Optional flavor specification. Can be used to load a particular flavor in case there are multiple flavors available.
run_id	Optional MLflow run-id. If supplied model will be fetched from MLflow tracking server.

mlflow_log_artifact *Log Artifact*

Description

Logs an specific file or directory as an artifact.

Usage

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

Arguments

path	The file or directory to log as an artifact.
artifact_path	Destination path within the run's artifact URI.

Details

The fluent API family of functions operate with an implied MLflow client determined by the service set by 'mlflow_set_tracking_uri()'. For operations involving a run it adopts the current active run, or, if one does not exist, starts one through the implied service.

When logging to Amazon S3, ensure that the user has a proper policy attach to it, for instance:

```
 { "Version": "2012-10-17","Statement": [  {    "Sid": "VisualEditor0",    "Effect": "Allow",    "Acti
"s3:PutObject",     "s3:GetObject",     "s3:ListBucket",     "s3:GetBucketLocation"
],   "Resource": [      "arn:aws:s3:::mlflow-test/*",     "arn:aws:s3:::mlflow-test"
]   }   ] }
```

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_metric *Log Metric*

Description

API to log a metric for a run. Metrics key-value pair that record a single float measure. During a single execution of a run, a particular metric can be logged several times. Backend will keep track of historical values along with timestamps.

Usage

```
mlflow_log_metric(key, value, timestamp = NULL)
```

Arguments

`key`	Name of the metric.
`value`	Float value for the metric being logged.
`timestamp`	Unix timestamp in milliseconds at the time metric was logged.

Details

The fluent API family of functions operate with an implied MLflow client determined by the service set by 'mlflow_set_tracking_uri()'. For operations involving a run it adopts the current active run, or, if one does not exist, starts one through the implied service.

mlflow_log_model *Log Model*

Description

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

Usage

```
mlflow_log_model(fn, artifact_path)
```

Arguments

fn	The serving function that will perform a prediction.
artifact_path	Destination path where this MLflow compatible model will be saved.

mlflow_log_param *Log Parameter*

Description

API to log a parameter used for this run. Examples are params and hyperparams used for ML training, or constant dates and values used in an ETL pipeline. A params 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)
```

Arguments

key	Name of the parameter.
value	String value of the parameter.

Details

The fluent API family of functions operate with an implied MLflow client determined by the service set by 'mlflow_set_tracking_uri()'. For operations involving a run it adopts the current active run, or, if one does not exist, starts one through the implied service.

mlflow_param *Read Command Line Parameter*

Description

Reads a command line parameter.

Usage

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

Arguments

name	The name for this parameter.
default	The default value for this 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 this parameter.

mlflow_predict_flavor *Predict over MLflow Model Flavor*

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_flavor(model, data)
```

Arguments

model	The loaded MLflow model flavor.
data	A data frame to perform scoring.

mlflow_predict_model *Generate prediction with MLflow model.*

Description

Generate prediction with MLflow model.

Usage

```
mlflow_predict_model(model, data)
```

Arguments

model	MLflow model.
data	Dataframe to be scored.

mlflow_restore_snapshot *Restore Snapshot*

Description

Restores a snapshot of all dependencies required to run the files in the current directory

Usage

```
mlflow_restore_snapshot()
```

mlflow_rfunc_predict *Predict using RFunc MLflow Model*

Description

Predict using an RFunc MLflow Model from a file or data frame.

Usage

```
mlflow_rfunc_predict(model_path, run_uuid = NULL, input_path = NULL,
  output_path = NULL, data = NULL, restore = FALSE)
```

Arguments

model_path	The path to the MLflow model, as a string.
run_uuid	Run ID of run to grab the model from.
input_path	Path to 'JSON' or 'CSV' file to be used for prediction.
output_path	'JSON' or 'CSV' file where the prediction will be written to.
data	Data frame to be scored. This can be utilized for testing purposes and can only be specified when 'input_path' is not specified.
restore	Should `mlflow_restore_snapshot()` be called before serving?

Examples

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

# save simple model which roundtrips data as prediction
mlflow_save_model(function(df) df, "mlflow_roundtrip")

# save data as json
jsonlite::write_json(iris, "iris.json")

# predict existing model from json data
mlflow_rfunc_predict("mlflow_roundtrip", "iris.json")

## End(Not run)
```

---

mlflow_rfunc_serve *Serve an RFunc MLflow Model*

Description

Serve an RFunc MLflow Model as a local web api.

Usage

```
mlflow_rfunc_serve(model_path, run_uuid = NULL, host = "127.0.0.1",
  port = 8090, daemonized = FALSE, browse = !daemonized,
  restore = FALSE)
```

Arguments

model_path	The path to the MLflow model, as a string.
run_uuid	ID of run to grab the model from.
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?
restore	Should `mlflow_restore_snapshot()` be called before serving?

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 in MLflow*

Description

Wrapper for 'mlflow run'.

Usage

```
mlflow_run(entry_point = NULL, uri = ".", version = NULL,
  param_list = NULL, experiment_id = NULL, mode = NULL,
  cluster_spec = NULL, git_username = NULL, git_password = NULL,
  no_conda = FALSE, storage_dir = NULL)
```

Arguments

entry_point	Entry point within project, defaults to 'main' if not specified.
uri	A directory containing modeling scripts, defaults to the current directory.
version	Version of the project to run, as a Git commit reference for Git projects.
param_list	A list of parameters.
experiment_id	ID of the experiment under which to launch the run.
mode	Execution mode to use for run.
cluster_spec	Path to JSON file describing the cluster to use when launching a run on Databricks.
git_username	Username for HTTP(S) Git authentication.

git_password	Password for HTTP(S) Git authentication.
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	Only valid when 'mode' 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.

`mlflow_save_flavor` *Save MLflow Model Flavor*

Description

Saves model in MLflow's flavor, to be used by package authors to extend the supported MLflow models.

Usage

```
mlflow_save_flavor(x, path = "model", r_dependencies = NULL,
  conda_env = NULL)
```

Arguments

x	The serving function or model that will perform a prediction.
path	Destination path where this MLflow compatible model will be saved.
r_dependencies	Optional vector of paths to dependency files to include in the model, as in `r-dependencies.txt` or `conda.yaml`.
conda_env	Path to Conda dependencies file.

Value

This funciton must return a list of flavors that conform to the MLmodel specification.

mlflow_save_flavor.keras.engine.training.Model *Save MLflow Keras Model Flavor*

Description

Saves model in MLflow's Keras flavor.

Usage

```
## S3 method for class 'keras.engine.training.Model'
mlflow_save_flavor(x,
  path = "model", r_dependencies = NULL, conda_env = NULL)
```

Arguments

x	The serving function or model that will perform a prediction.
path	Destination path where this MLflow compatible model will be saved.
r_dependencies	Optional vector of paths to dependency files to include in the model, as in `r-dependencies.txt` or `conda.yaml`.
conda_env	Path to Conda dependencies file.

Value

This funciton must return a list of flavors that conform to the MLmodel specification.

mlflow_save_model *Save Model for MLflow*

Description

Saves model in MLflow's format that can later be used for prediction and serving.

Usage

```
mlflow_save_model(x, path = "model", r_dependencies = NULL,
  conda_env = NULL)
```

Arguments

x	The serving function or model that will perform a prediction.
path	Destination path where this MLflow compatible model will be saved.
r_dependencies	Optional vector of paths to dependency files to include in the model, as in `r-dependencies.txt` or `conda.yaml`.
conda_env	Path to Conda dependencies file.

mlflow_server *Run the 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

Set given experiment as active experiment. If experiment does not exist, create an experiment with provided name.

Usage

```
mlflow_set_experiment(experiment_name)
```

Arguments

experiment_name	Name of experiment to be activated.

Details

The fluent API family of functions operate with an implied MLflow client determined by the service set by 'mlflow_set_tracking_uri()'. For operations involving a run it adopts the current active run, or, if one does not exist, starts one through the implied service.

mlflow_set_tag *Set Tag*

Description

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

Usage

```
mlflow_set_tag(key, value)
```

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.

Details

The fluent API family of functions operate with an implied MLflow client determined by the service set by ‘mlflow_set_tracking_uri()‘. For operations involving a run it adopts the current active run, or, if one does not exist, starts one through the implied service.

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_snapshot *Dependencies Snapshot*

Description

Creates a snapshot of all dependencies required to run the files in the current directory.

Usage

```
mlflow_snapshot()
```

mlflow_start_run *Start Run*

Description

Starts a new run within an experiment, should be used within a `with` block.

Usage

```
mlflow_start_run(run_uuid = NULL, experiment_id = NULL,
  source_name = NULL, source_version = NULL, entry_point_name = NULL,
  source_type = "LOCAL")
```

Arguments

`run_uuid`	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_uuid" 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.
`source_name`	Name of the source file or URI of the project to be associated with the run. Defaults to the current file if none provided.
`source_version`	Optional Git commit hash to associate with the run.
`entry_point_name`	Optional name of the entry point for to the current run.
`source_type`	Integer enum value describing the type of the run ("local", "project", etc.).

Details

The fluent API family of functions operate with an implied MLflow client determined by the service set by 'mlflow_set_tracking_uri()'. For operations involving a run it adopts the current active run, or, if one does not exist, starts one through the implied service.

Examples

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

## End(Not run)
```

`mlflow_ui` *MLflow User Interface*

Description

Launches MLflow user interface.

Usage

```
mlflow_ui(x, ...)
```

Arguments

`x`	An ‘mlflow_client‘ object.
`...`	Optional arguments passed to ‘mlflow_server()‘ when ‘x‘ is a path to a file store.

Examples

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

# 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_uninstall` *Uninstalls MLflow.*

Description

Uninstalls MLflow by removing the Conda environment.

Usage

```
mlflow_uninstall()
```

Examples

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

## End(Not run)
```

Index