Package 'pysparklyr'

May 19, 2025

Title Provides a 'PySpark' Back-End for the 'sparklyr' Package

Version 0.1.8

Description It enables 'sparklyr' to integrate with 'Spark Connect', and 'Databricks Connect' by providing a wrapper over the 'PySpark' 'python' library.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Imports arrow, cli, DBI, dplyr, dbplyr, glue, purrr, reticulate (>= 1.41.0.1), methods, rlang, sparklyr (>= 1.9.0), tidyselect, fs, magrittr, tidyr, vctrs, processx, httr2, rstudioapi, rsconnect

URL https://github.com/mlverse/pysparklyr

BugReports https://github.com/mlverse/pysparklyr/issues

Suggests crayon, R6, testthat (>= 3.0.0), tibble, withr

Config/testthat/edition 3

NeedsCompilation no

Author Edgar Ruiz [aut, cre], Posit Software, PBC [cph, fnd]

Maintainer Edgar Ruiz <edgar@posit.co>

Repository CRAN

Date/Publication 2025-05-19 20:50:02 UTC

Contents

`deploy_databricks` *Deploys Databricks backed content to publishing server*

Description

This is a convenience function that is meant to make it easier for you to publish your Databricks backed content to a publishing server. It is meant to be primarily used with Posit Connect.

Usage

```
deploy_databricks(
  appDir = NULL,
  python = NULL,
  account = NULL,
  server = NULL,
  lint = FALSE,
  forceGeneratePythonEnvironment = TRUE,
  version = NULL,
  cluster_id = NULL,
  host = NULL,
  token = NULL,
  confirm = interactive(),
  ...
)
```

Arguments

`appDir`	A directory containing an application (e.g. a Shiny app or plumber API) Defaults to NULL. If left NULL, and if called within RStudio, it will attempt to use the folder of the currently opened document within the IDE. If there are no opened documents, or not working in the RStudio IDE, then it will use `getwd()` as the default value.
`python`	Full path to a python binary for use by `reticulate`. It defaults to NULL. If left NULL, this function will attempt to find a viable local Python environment to replicate using the following hierarchy: 1. `version` - Cluster's DBR version 2. `cluster_id` - Query the cluster to obtain its DBR version 3. If one is loaded in the current R session, it will verify that the Python environment is suited to be used as the one to use
`account`	The name of the account to use to publish
`server`	The name of the target server to publish
`lint`	Lint the project before initiating the project? Default to FALSE. It has been causing issues for this type of content.

forceGeneratePythonEnvironment
: If an existing requirements.txt file is found, it will be overwritten when this argument is TRUE.

version
: The Databricks Runtime (DBR) version. Use if python is NULL.

cluster_id
: The Databricks cluster ID. Use if python, and version are NULL

host
: The Databricks host URL. Defaults to NULL. If left NULL, it will use the environment variable DATABRICKS_HOST

token
: The Databricks authentication token. Defaults to NULL. If left NULL, it will use the environment variable DATABRICKS_TOKEN

confirm
: Should the user be prompted to confirm that the correct information is being used for deployment? Defaults to interactive()

...
: Additional named arguments passed to rsconnect::deployApp() function

Value

No value is returned to R. Only output to the console.

---

installed_components *Lists installed Python libraries*

Description

Lists installed Python libraries

Usage

```
installed_components(list_all = FALSE)
```

Arguments

list_all
: Flag that indicates to display all of the installed packages or only the top two, namely, pyspark and databricks.connect

Value

Returns no value, only sends information to the console. The information includes the current versions of 'sparklyr', and 'pysparklyr', as well as the 'Python' environment currently loaded.

`install_pyspark` *Installs PySpark and Python dependencies*

Description

Installs PySpark and Python dependencies

Installs Databricks Connect and Python dependencies

Usage

```
install_pyspark(
  version = NULL,
  envname = NULL,
  python_version = NULL,
  new_env = TRUE,
  method = c("auto", "virtualenv", "conda"),
  as_job = TRUE,
  install_ml = FALSE,
  ...
)

install_databricks(
  version = NULL,
  cluster_id = NULL,
  envname = NULL,
  python_version = NULL,
  new_env = TRUE,
  method = c("auto", "virtualenv", "conda"),
  as_job = TRUE,
  install_ml = FALSE,
  ...
)
```

Arguments

`version`	Version of 'databricks.connect' to install. Defaults to `NULL`. If `NULL`, it will check against PyPi to get the current library version.
`envname`	The name of the Python Environment to use to install the Python libraries. Defaults to `NULL`. If `NULL`, a name will automatically be assigned based on the version that will be installed
`python_version`	The minimum required version of Python to use to create the Python environment. Defaults to `NULL`. If `NULL`, it will check against PyPi to get the minimum required Python version.
`new_env`	If `TRUE`, any existing Python virtual environment and/or Conda environment specified by `envname` is deleted first.

method	The installation method to use. If creating a new environment, "auto" (the default) is equivalent to "virtualenv". Otherwise "auto" infers the installation method based on the type of Python environment specified by envname.
as_job	Runs the installation if using this function within the RStudio IDE.
install_ml	Installs ML related Python libraries. Defaults to TRUE. This is mainly for machines with limited storage to avoid installing the rather large 'torch' library if the ML features are not going to be used. This will apply to any environment backed by 'Spark' version 3.5 or above.
...	Passed on to reticulate::py_install()
cluster_id	Target of the cluster ID that will be used with. If provided, this value will be used to extract the cluster's version

Value

It returns no value to the R session. This function purpose is to create the 'Python' environment, and install the appropriate set of 'Python' libraries inside the new environment. During runtime, this function will send messages to the console describing the steps that the function is taking. For example, it will let the user know if it is getting the latest version of the Python library from 'PyPi.org', and the result of such query.

---

ml_prepare_dataset *Creates the 'label' and 'features' columns*

Description

Creates the 'label' and 'features' columns

Usage

```
ml_prepare_dataset(
  x,
  formula = NULL,
  label = NULL,
  features = NULL,
  label_col = "label",
  features_col = "features",
  keep_original = TRUE,
  ...
)
```

Arguments

x	A tbl_pyspark object
formula	Used when x is a tbl_spark. R formula.
label	The name of the label column.

features	The name(s) of the feature columns as a character vector.
label_col	Label column name, as a length-one character vector.
features_col	Features column name, as a length-one character vector.
keep_original	Boolean flag that indicates if the output will contain, or not, the original columns from x. Defaults to TRUE.
...	Added for backwards compatibility. Not in use today.

Details

At this time, 'Spark ML Connect', does not include a Vector Assembler transformer. The main thing that this function does, is create a 'Pyspark' array column. Pipelines require a 'label' and 'features' columns. Even though it is is single column in the dataset, the 'features' column will contain all of the predictors insde an array. This function also creates a new 'label' column that copies the outcome variable. This makes it a lot easier to remove the 'label', and 'outcome' columns.

Value

A tbl_pyspark, with either the original columns from x, plus the 'label' and 'features' column, or, the 'label' and 'features' columns only.

---

pyspark_config *Read Spark configuration*

Description

Read Spark configuration

Usage

```
pyspark_config()
```

Value

A list object with the initial configuration that will be used for the Connect session.

`requirements_write` *Writes the 'requirements.txt' file, containing the needed Python libraries*

Description

This is a helper function that it is meant to be used for deployments of the document or application. By default, `deploy_databricks()` will run this function the first time you use that function to deploy content to Posit Connect.

Usage

```
requirements_write(
  envname = NULL,
  destfile = "requirements.txt",
  overwrite = FALSE,
  ...
)
```

Arguments

`envname`	The name of, or path to, a Python virtual environment.
`destfile`	Target path for the requirements file. Defaults to 'requirements.txt'.
`overwrite`	Replace the contents of the file if it already exists?
`...`	Additional arguments passed to `reticulate::py_list_packages()`

Value

No value is returned to R. The output is a text file with the list of Python libraries.

`spark_connect_service_start` *Starts and stops Spark Connect locally*

Description

Starts and stops Spark Connect locally

Usage

```
spark_connect_service_start(
  version = "3.5",
  scala_version = "2.12",
  include_args = TRUE,
  ...
)

spark_connect_service_stop(version = "3.5", ...)
```

Arguments

version	Spark version to use (3.4 or above)
scala_version	Acceptable Scala version of packages to be loaded
include_args	Flag that indicates whether to add the additional arguments to the command that starts the service. At this time, only the 'packages' argument is submitted.
...	Optional arguments; currently unused

Value

It returns messages to the console with the status of starting, and stopping the local Spark Connect service.

Index