Package ‘pysparklyr’

December 6, 2023

Title Provides a 'PySpark' Back-End for the 'sparklyr' Package
Version 0.1.2
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 7.2.3
Imports cli, DBI, dplyr, dbplyr, glue, purrr, reticulate (>= 1.33),
methods, rlang, sparklyr (>= 1.8.4), tidyselect, fs, magrittr,
tidyrr, vctrs, processx, httr2, rstudioapi
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 2023-12-06 00:40:02 UTC

R topics documented:

  installed_components .......................................................... 2
  install_pyspark ................................................................. 2
  ml_prepare_dataset ........................................................... 3
  spark_connect_service_start ............................................. 4

Index 6
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 = ">=3.9",
    new_env = TRUE,
    method = c("auto", "virtualenv", "conda"),
    as_job = TRUE,
    install_ml = FALSE,
    ...
)

install_databricks(
    version = NULL,
    cluster_id = NULL,
ml_prepare_dataset

```
envname = NULL,  
python_version = ">=3.9",  
new_env = TRUE,  
method = c("auto", "virtualenv", "conda"),  
as_job = TRUE,  
install_ml = FALSE,  
...  
)
```

**Arguments**

- **version**: Version of 'databricks.connect' to install
- **envname**: The name of the Python Environment to use to install the Python libraries. Default to NULL. If NULL, a name will automatically be assigned based on the version that will be installed.
- **python_version**: The version of Python to use to create the Python environment.
- **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

```r
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 inside 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.

Description

Starts and stops Spark Connect locally
spark_connect_service_start

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

install_databricks (install_pyspark), 2
install_pyspark, 2
installed_components, 2
ml_prepare_dataset, 3
reticulate::py_install(), 3
spark_connect_service_start, 4
spark_connect_service_stop
   (spark_connect_service_start), 4