Package 'featuretoolsR'

August 7, 2019

Type Package

Title Interact with the 'Python' Module 'Featuretools'

Version 0.4.3

Maintainer Magnus Furugård <magnus.furugard@gmail.com>

Description A 'reticulate'-based interface to the 'Python' module 'Featuretools'.
The package grants functionality to interact with 'Pythons' 'Featuretools' module, which allows for automated feature engineering on any data frame. Valid features and new data sets can, after feature synthesis, easily be extracted.

License MIT + file LICENSE

URL https://github.com/magnusfurugard/featuretoolsR

BugReports https://github.com/magnusfurugard/featuretoolsR/issues

Depends R (>= 3.4.2)

Imports reticulate, caret, dplyr, purrr, stringr, tibble, magrittr, cli, testthat

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

NeedsCompilation no

Author Magnus Furugård [aut, cre]

Repository CRAN

Date/Publication 2019-08-07 08:40:02 UTC

R topics documented:

add_entity 2
add_relationship 3
as_entityset 4
calculate_feature_matrix 4
create_entityset 5
dfs 6

extract_features 7
install_featuretools 8
list_primitives 8
load_features 9
save_features 10
tidy_feature_matrix 10

Index **12**

`add_entity` *add_entity*

Description

Add an entity to an entityset.

Usage

```
add_entity(entityset, entity_id, df, index = NULL, time_index = NULL,
  ...)
```

Arguments

`entityset`	The entity set to modify.
`entity_id`	The name of the entity to add.
`df`	The data frame to add as an entity.
`index`	The index parameter specifies the column that uniquely identifies rows in the dataframe
`time_index`	Name of the time column in the dataframe.
`...`	Additional parameters passed to 'featuretools.entity_from_dataframe'.

Value

A modified entityset.

Examples

```
library(magrittr)
create_entityset("set") %>%
  add_entity(df = cars,
             entity_id = "cars",
             index = "row_number")
```

`add_relationship` *Add a relationship to an entityset*

Description

Add a relationship to an entityset.

Usage

```
add_relationship(entityset, parent_set, child_set, parent_idx, child_idx)
```

Arguments

`entityset`	The entityset to modify.
`parent_set`	The name of the parent set.
`child_set`	The name of the child set.
`parent_idx`	The index variable of the ‘parent_set‘.
`child_idx`	The index variable of the ‘child_set‘.

Value

A modified entityset.

Examples

```
library(magrittr)
set_1 <- data.frame(key = 1:100, value = sample(letters, 100, TRUE), stringsAsFactors = TRUE)
set_2 <- data.frame(key = 1:100, value = sample(LETTERS, 100, TRUE), stringsAsFactors = TRUE)
# Common variable: `key`

as_entityset(set_1, index = "key", entity_id = "set_1", id = "demo") %>%
  add_entity(entity_id = "set_2", df = set_2, index = "key") %>%
  add_relationship(
    parent_set = "set_1",
    child_set = "set_2",
    parent_idx = "key",
    child_idx = "key"
  )
```

as_entityset *Create entityset and entity from data frame.*

Description

Create an entityset with a selected ‘data.frame‘ as an entity.

Usage

```
as_entityset(.data, id = "entityset", index = NA, time_index = NULL,
  entity_id = "df1", ...)
```

Arguments

.data	The ‘data.frame‘ to be added as an entity to entityset.
id	The id of this entityset.
index	Name of id column in the dataframe.
time_index	Name of the time column in the dataframe.
entity_id	An identifier for this entity.
...	Additional variables passed to ‘add_entity‘.

Value

A modified entityset.

Examples

```
as_entityset(cars, index = "row_number")
```

calculate_feature_matrix *Calculate feature matrix*

Description

This function is used to create a feature matrix based on a custom list of features (usually created from save_features).

Usage

```
calculate_feature_matrix(entityset, features, ...)
```

Arguments

entityset	The entityset on which to create features.
features	The features to create based on previous runs of dfs.
...	Additional parameters passed to ‘featuretoools.calculate_feature_matrix‘.

Value

A feature matrix

Examples

```
library(magrittr)

# Create some mock data
set_1 <- data.frame(key = 1:100, value = sample(letters, 100, TRUE), stringsAsFactors = TRUE)
set_2 <- data.frame(key = 1:100, value = sample(LETTERS, 100, TRUE), stringsAsFactors = TRUE)
# Common variable: `key`

# Create features and save them
as_entityset(set_1, index = "key", entity_id = "set_1", id = "demo") %>%
  add_entity(entity_id = "set_2", df = set_2, index = "key") %>%
  add_relationship(
    parent_set = "set_1",
    child_set = "set_2",
    parent_idx = "key",
    child_idx = "key"
  ) %>%
  dfs(target_entity = "set_1", trans_primitives = c("and")) %>%
  extract_features() %>%
  save_features(filename = "some.features")

# Re-create entityset, but rather than dfs use calcualte_feature_matrix.
es <- as_entityset(set_1, index = "key", entity_id = "set_1", id = "demo") %>%
  add_entity(entity_id = "set_2", df = set_2, index = "key") %>%
  add_relationship(
    parent_set = "set_1",
    child_set = "set_2",
    parent_idx = "key",
    child_idx = "key"
  )
calculate_feature_matrix(entityset = es, features = load_features("some.features"))
```

---

create_entityset *Create entityset*

Description

Create a blank entityset. A shortcut for ‘featuretools’‘ ‘EntitySet‘.

Usage

```
create_entityset(id)
```

Arguments

id	The id of this entityset.

Value

An entityset.

Examples

```
create_entityset(id = "my_entityset")
```

---

dfs *Deep Feature Synthesis*

Description

The main function from featuretools used to create new features.

Usage

```
dfs(entityset, target_entity, agg_primitives = NULL,
  trans_primitives = NULL, max_depth = 2L, ...)
```

Arguments

entityset	The entityset on which to perform dfs.
target_entity	The name of the entity on which to perform dfs.
agg_primitives	Primitives passed to relational data.
trans_primitives	Primitives passed to non-relational data.
max_depth	Controls the maximum depth of features.
...	Additional parameters passed to ‘featuretools.dfs‘.

Value

A ‘featuretools‘ feature matrix.

Examples

```
es <- as_entityset(cars, index = "row_number")
dfs(es, target_entity = "df1", trans_primitives = c("and"))
```

`extract_features` *Extract features*

Description

This function is used to extract all features created from dfs.

Usage

```
extract_features(.data)
```

Arguments

`.data`	The featuretools-object returned from dfs.

Value

All features created during dfs, as a tibble.

Examples

```
library(magrittr)
set_1 <- data.frame(key = 1:100, value = sample(letters, 100, TRUE), stringsAsFactors = TRUE)
set_2 <- data.frame(key = 1:100, value = sample(LETTERS, 100, TRUE), stringsAsFactors = TRUE)
# Common variable: `key`

as_entityset(set_1, index = "key", entity_id = "set_1", id = "demo") %>%
  add_entity(entity_id = "set_2", df = set_2, index = "key") %>%
  add_relationship(
    parent_set = "set_1",
    child_set = "set_2",
    parent_idx = "key",
    child_idx = "key"
  ) %>%
  dfs(target_entity = "set_1", trans_primitives = c("and")) %>%
  extract_features()
```

install_featuretools *Install featuretools*

Description

Setup for featuretools in it's own virtualenv, or into the default reticulate virtualenv.

Usage

```
install_featuretools(custom_virtualenv = FALSE, method = "auto",
  conda = "auto")
```

Arguments

custom_virtualenv	Defaults to false. Set to true if you wish to use a custom virtualenv for featuretoolsR.
method	The installation method passed to 'reticulate::py_install'. Defaults to "auto".
conda	Whether to use conda or not. Passed to 'reticulate::py_install'. Defaults to "auto".

Examples

```
## Not run:
featuretoolsR::install_featuretools()

## End(Not run)
```

list_primitives *List all available primitives.*

Description

List all available primitives from 'featuretools' which can be passed to dfs.

Usage

```
list_primitives()
```

Value

A list of all primitives available.

Examples

```
featuretoolsR::list_primitives()
```

load_features *Load features*

Description

Used to load previously saved features created during dfs.

Usage

```
load_features(file = NA)
```

Arguments

file	The file containing the features.

Examples

```
library(magrittr)

# Create mock datasets
set_1 <- data.frame(key = 1:100, value = sample(letters, 100, TRUE), stringsAsFactors = TRUE)
set_2 <- data.frame(key = 1:100, value = sample(LETTERS, 100, TRUE), stringsAsFactors = TRUE)
# Common variable: `key`

# Use dfs to create features
dir <- tempdir()
as_entityset(set_1, index = "key", entity_id = "set_1", id = "demo") %>%
  add_entity(entity_id = "set_2", df = set_2, index = "key") %>%
  add_relationship(
    parent_set = "set_1",
    child_set = "set_2",
    parent_idx = "key",
    child_idx = "key"
  ) %>%
  dfs(target_entity = "set_1", trans_primitives = c("and")) %>%
  extract_features() %>%
  save_features(filename = "some.features", path = dir)

# Load saves features
features <- load_features(file.path(dir, "some.features"))
```

save_features *Save features*

Description

Used to save all or a subset of features created during dfs.

Usage

```
save_features(.data, filename = NA, path = NA)
```

Arguments

.data	The tibble of features returned from extract_features.
filename	(optional) The name of the file to produce.
path	(optional) The path where the feature file should be placed.

Examples

```
library(magrittr)
set_1 <- data.frame(key = 1:100, value = sample(letters, 100, TRUE), stringsAsFactors = TRUE)
set_2 <- data.frame(key = 1:100, value = sample(LETTERS, 100, TRUE), stringsAsFactors = TRUE)
# Common variable: `key`

dir <- tempdir()
as_entityset(set_1, index = "key", entity_id = "set_1", id = "demo") %>%
  add_entity(entity_id = "set_2", df = set_2, index = "key") %>%
  add_relationship(
    parent_set = "set_1",
    child_set = "set_2",
    parent_idx = "key",
    child_idx = "key"
  ) %>%
  dfs(target_entity = "set_1", trans_primitives = c("and")) %>%
  extract_features() %>%
  save_features(filename = "some.features", path = dir)
```

tidy_feature_matrix *Tidy feature matrix*

Description

Used for tidying up ('R-ify') the feature matrix after deep feature synthethis (dfs).

Usage

```
tidy_feature_matrix(.data, remove_nzv = FALSE, nan_is_na = FALSE,
  clean_names = FALSE)
```

Arguments

.data	The featuretools-object returned from dfs.
remove_nzv	Remove near zero variance variables created from dfs.
nan_is_na	Turn all 'NaN' into 'NA'.
clean_names	Make variable names R-friendly (snake case).

Value

A tidy data.frame.

Examples

```
library(magrittr)
set_1 <- data.frame(key = 1:100, value = sample(letters, 100, TRUE), stringsAsFactors = TRUE)
set_2 <- data.frame(key = 1:100, value = sample(LETTERS, 100, TRUE), stringsAsFactors = TRUE)
# Common variable: `key`

as_entityset(set_1, index = "key", entity_id = "set_1", id = "demo") %>%
  add_entity(entity_id = "set_2", df = set_2, index = "key") %>%
  add_relationship(
    parent_set = "set_1",
    child_set = "set_2",
    parent_idx = "key",
    child_idx = "key"
  ) %>%
  dfs(target_entity = "set_1", trans_primitives = c("and")) %>%
  tidy_feature_matrix(remove_nzv = TRUE, nan_is_na = TRUE)
```

Index

add_entity, 2
add_relationship, 3
as_entityset, 4

calculate_feature_matrix, 4
create_entityset, 5

dfs, *5*, 6, *7–11*

extract_features, 7, *10*

install_featuretools, 8

list_primitives, 8
load_features, 9

save_features, *4*, 10

tidy_feature_matrix, 10