Package 'pipeliner'

May 9, 2026

Type Package

Version 0.1.1

Title Machine Learning Pipelines for R

Date 2016-12-16

Author Alex Ioannides

Maintainer Alex Ioannides <alex.ioannides@yahoo.co.uk>

Description A framework for defining 'pipelines' of functions for applying data transformations, model estimation and inverse-transformations, resulting in predicted value generation (or model-scoring) functions that automatically apply the entire pipeline of functions required to go from input to predicted output.

License Apache License 2.0

URL https://github.com/alexioannides/pipeliner

BugReports https://github.com/alexioannides/pipeliner/issues

LazyData TRUE

Suggests knitr, testthat, rmarkdown, modelr, tidyverse

VignetteBuilder knitr

RoxygenNote 5.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2016-12-19 15:56:03

Contents

`cbind_fast` *Faster alternative to* `cbind_fast`

Description

This is not as 'safe' as using `cbind_fast` - for example, if `df1` has columns with the same name as columns in `df2`, then they will be over-written.

Usage

```
cbind_fast(df1, df2)
```

Arguments

`df1`	A data.frame.
`df2`	Another data.frame

Value

A data.frame equal to `df1` with the columns of `df2` appended.

Examples

```
## Not run:
df1 <- data.frame(x = 1:5, y = 1:5 * 0.1)
df2 <- data.frame(a = 6:10, b = 6:10 * 0.25)
df3 <- cbind_fast(df1, df2)
df3
#   x   y  a    b
# 1 1 0.1  6 1.50
# 2 2 0.2  7 1.75
# 3 3 0.3  8 2.00
# 4 4 0.4  9 2.25
# 5 5 0.5 10 2.50

## End(Not run)
```

check_data_frame_throw_error

Validate ml_pipeline_builder transform method returns data.frame

Description

Helper function that checks if the object returned from a `ml_pipeline_builder` method is data.frame (if it isn't NULL), and if it isn't, throws an error that is customised with the returning name.

Usage

```
check_data_frame_throw_error(func_return_object, func_name)
```

Arguments

`func_return_object`	The object returned from a `ml_pipeline_builder` method.
`func_name`	The name of the function that returned the object.

Examples

```
## Not run:
transform_method <- function(df) df
data <- data.frame(y = c(1, 2), x = c(0.1, 0.2))
data_transformed <- transform_method(data)
check_data_frame_throw_error(data_transformed, "transform_method")
# NULL

## End(Not run)
```

check_predict_method_throw_error

Validate estimate_model method returns an object with a `predict` *method defined*

Description

Helper function that checks if the object returned from the `estimate_model` method has a `predict` method defined for it.

Usage

```
check_predict_method_throw_error(func_return_object)
```

Arguments

func_return_object	The object returned from the estimate_model method.

Examples

```
## Not run:
estimation_method <- function(df) lm(eruptions ~ 0 + waiting, df)
data <- faithful
model_estimate <- estimation_method(data)
check_predict_method_throw_error(model_estimate)
# NULL

## End(Not run)
```

---

check_unary_func_throw_error

Validate ml_pipeline_builder transform method is a unary function

---

Description

Helper function that checks if a ml_pipeline_builder method is unary function (if it isn't a NULL returning function), and if it isn't, throws an error that is customised with the method function name.

Usage

```
check_unary_func_throw_error(func, func_name)
```

Arguments

func	A ml_pipeline_builder method.
func_name	The name of the ml_pipeline_builder method.

Examples

```
## Not run:
transform_method <- function(df) df
check_unary_func_throw_error(transform_method, "transform_method")
# NULL

## End(Not run)
```

estimate_model *Estimate machine learning model*

Description

A function that takes as its arguement another function defining how a machine learning model should be estimated based on the variables available in the input data frame. This function is wrapped (or adapted) for use within a machine learning pipeline.

Usage

```
estimate_model(.f)
```

Arguments

`.f`	A unary function of a data.frame that returns a fitted model object, which must have a `predict.{model-class}` defined and available in the enclosing environment. An error will be thrown if any of these criteria are not met.

Value

A unary function of a data.frame that returns a fitted model object that has a `predict.{model-class}` defined This function is assigned the classes `"estimate_model"` and `"ml_pipeline_section"`.

Examples

```
data <- head(faithful)
f <- estimate_model(function(df) {
  lm(eruptions ~ 1 + waiting, df)
})

f(data)
# Call:
#   lm(formula = eruptions ~ 1 + waiting, data = df)
#
# Coefficients:
# (Intercept)      waiting
#    -1.53317      0.06756
```

func_error_handler *Custom error handler for printing the name of an enclosing function with error*

Description

Custom error handler for printing the name of an enclosing function with error

Usage

```
func_error_handler(e, calling_func)
```

Arguments

e	A simpleError - e.g. thrown from tryCatch
calling_func	A character string naming the enclosing function (or closure) for printing with error messages

Value

NULL - throws error with custom message

Examples

```
## Not run:
f <- function(x) x ^ 2
tryCatch(f("a"), error = function(e) func_error_handler(e, "f"))
# Error in x^2 : non-numeric argument to binary operator
# ---> called from within function: f

## End(Not run)
```

---

inv_transform_response

Inverse transform machine learning response variable

Description

A function that takes as its arguement another function defining a inverse response variable transformation, and wraps (or adapts) it for use within a machine learning pipeline.

Usage

```
inv_transform_response(.f)
```

Arguments

.f	A unary function of a data.frame that returns a new data.frame containing only the inverse transformed response variable. An error will be thrown if this is not the case.

Value

A unary function of a data.frame that returns the input data.frame with the inverse transformed response variable column appended. This function is assigned the classes "inv_transform_response" and "ml_pipeline_section".

Examples

```
data <- head(faithful)
f1 <- transform_response(function(df) {
  data.frame(y = (df$eruptions - mean(df$eruptions)) / sd(df$eruptions))
})
f2 <- inv_transform_response(function(df) {
  data.frame(eruptions2 = df$y * sd(df$eruptions) + mean(df$eruptions))
})

f2(f1(data))
#   eruptions waiting          y eruptions2
# 1     3.600      79  0.5412808      3.600
# 2     1.800      54 -1.3039946      1.800
# 3     3.333      74  0.2675649      3.333
# 4     2.283      62 -0.8088457      2.283
# 5     4.533      85  1.4977485      4.533
# 6     2.883      55 -0.1937539      2.883
```

`ml_pipline_builder` *Build machine learning pipelines - object oriented API*

Description

Building machine learning models often requires pre- and post-transformation of the input and/or response variables, prior to training (or fitting) the models. For example, a model may require training on the logarithm of the response and input variables. As a consequence, fitting and then generating predictions from these models requires repeated application of transformation and inverse-transormation functions, to go from the original input to original output variables (via the model).

Usage

```
ml_pipline_builder()
```

Details

This function produces an object in which it is possible to: define transformation and inverse-transformation functions; fit a model on training data; and then generate a prediction (or model-scoring) function that automatically applies the entire pipeline of transformation and inverse-transformation to the inputs and outputs of the inner-model's predicted scores.

Calling `ml_pipline_builder()` will return an 'ml_pipeline' object (actually an environment or closure), whose methods can be accessed as one would access any element of a list. For example, `ml_pipline_builder()$transform_features` will allow you to get or set the `transform_features` function to use the pipeline. The full list of methods for defining sections of the pipeline (documented elsewhere) are:

- `transform_features`;
- `transform_response`;

- `inv_transform_response`; and,
- `estimate_model`;

The pipeline can be fit, prediction generated and the inner model accessed using the following methods:

- `fit(.data)`;
- `predict(.data)`; and,
- `model_estimate()`.

Value

An object of class `ml_pipeline`.

See Also

`transform_features`, `transform_response`, `estimate_model` and `inv_transform_response`.

Examples

```
data <- faithful

lm_pipeline <- ml_pipline_builder()

lm_pipeline$transform_features(function(df) {
  data.frame(x1 = (df$waiting - mean(df$waiting)) / sd(df$waiting))
})

lm_pipeline$transform_response(function(df) {
  data.frame(y = (df$eruptions - mean(df$eruptions)) / sd(df$eruptions))
})

lm_pipeline$inv_transform_response(function(df) {
  data.frame(pred_eruptions = df$pred_model * sd(df$eruptions) + mean(df$eruptions))
})

lm_pipeline$estimate_model(function(df) {
  lm(y ~ 0 + x1, df)
})

lm_pipeline$fit(data)
head(lm_pipeline$predict(data))
#    eruptions waiting         x1 pred_model pred_eruptions
#  1     3.600      79  0.5960248  0.5369058       4.100592
#  2     1.800      54 -1.2428901 -1.1196093       2.209893
#  3     3.333      74  0.2282418  0.2056028       3.722452
#  4     2.283      62 -0.6544374 -0.5895245       2.814917
#  5     4.533      85  1.0373644  0.9344694       4.554360
#  6     2.883      55 -1.1693335 -1.0533487       2.285521
```

pipeline *Build machine learning pipelines - functional API*

Description

Building machine learning models often requires pre- and post-transformation of the input and/or response variables, prior to training (or fitting) the models. For example, a model may require training on the logarithm of the response and input variables. As a consequence, fitting and then generating predictions from these models requires repeated application of transformation and inverse-transormation functions, to go from the original input to original output variables (via the model).

Usage

```
pipeline(.data, ...)
```

Arguments

`.data`	A data.frame containing the input variables required to fit the pipeline.
`...`	Functions of class `"ml_pipeline_section"` - e.g. `transform_features()`, `transform_response()`, `inv_transform_response()` or `estimate_model()`.

Details

This function that takes individual pipeline sections - functions with class `"ml_pipeline_section"` - together with the data required to estimate the inner models, returning a machine pipeline capable of predicting (scoring) data end-to-end, without having to repeatedly apply input variable (feature and response) transformation and their inverses.

Value

A `"ml_pipeline"` object contaiing the pipeline prediction function `ml_pipeline$predict()` and the estimated machine learning model nested within it `ml_pipeline$inner_model()`.

Examples

```
data <- faithful

lm_pipeline <-
  pipeline(
    data,
    transform_features(function(df) {
      data.frame(x1 = (df$waiting - mean(df$waiting)) / sd(df$waiting))
    }),

    transform_response(function(df) {
      data.frame(y = (df$eruptions - mean(df$eruptions)) / sd(df$eruptions))
    }),
```

```
  estimate_model(function(df) {
    lm(y ~ 1 + x1, df)
  }),

  inv_transform_response(function(df) {
    data.frame(pred_eruptions = df$pred_model * sd(df$eruptions) + mean(df$eruptions))
  })
)
```

`pipeliner` *pipeliner: machine learning pipelines for R*

Description

Allows you to define, fit and predict machine learning pipelines.

`predict.ml_pipeline` *Predict method for ML pipelines*

Description

Predict method for ML pipelines

Usage

```
## S3 method for class 'ml_pipeline'
predict(object, data, verbose = FALSE,
  pred_var = "pred_model", ...)
```

Arguments

`object`	An estimated pipleine object of class `ml_pipeline`.
`data`	A data.frame in which to look for input variables with which to predict.
`verbose`	Boolean - whether or not to return data.frame with all input and interim variables as well as predictions.
`pred_var`	Name to assign to for column of predictions from the 'raw' (or inner) model in the pipeline.
`...`	Any additional arguements than need to be passed to the underlying model's predict methods.

Value

A vector of model predictions or scores (default); or, a data.frame containing the predicted values, input variables, as well as any interim tranformed variables.

Examples

```
data <- faithful

lm_pipeline <-
  pipeline(
    data,
    estimate_model(function(df) {
      lm(eruptions ~ 1 + waiting, df)
    })
  )

in_sample_predictions <- predict(lm_pipeline, data)
head(in_sample_predictions)
# [1] 4.100592 2.209893 3.722452 2.814917 4.554360 2.285521
```

`predict_model` *Generate machine learning model prediction*

Description

A helper function that takes as its arguement an estimated machine learning model and returns a prediction function for use within a machine learning pipeline.

Usage

```
predict_model(.m)
```

Arguments

`.m`	An estimated machine lerning model.

Value

A unary function of a data.frame that returns the input data.frame with the predicted response variable column appended. This function is assigned the classes `"predict_model"` and `"ml_pipeline_section"`.

Examples

```
## Not run:
data <- head(faithful)
m <- estimate_model(function(df) {
  lm(eruptions ~ 1 + waiting, df)
})

predict_model(m(data))(data, "pred_eruptions")
#   eruptions waiting pred_eruptions
# 1     3.600      79       3.803874
# 2     1.800      54       2.114934
# 3     3.333      74       3.466086
```

```
# 4     2.283      62       2.655395
# 5     4.533      85       4.209219
# 6     2.883      55       2.182492

## End(Not run)
```

`process_transform_throw_error` *Validate and clean transform function output*

Description

Helper function that ensures the output of applying a transform function is a data.frame and that this data frame does not duplicate variables from the original (input data) data frame. If duplicates are found they are automatically dropped from the data.frame that is returned by this function.

Usage

```
process_transform_throw_error(input_df, output_df, func_name)
```

Arguments

`input_df`	The original (input data) data.frame - the transform function's argument.
`output_df`	The the transform function's output.
`func_name`	The name of the `ml_pipeline_builder` trandform method.

Value

If the transform function is not `NULL` then a copy of the transform function's output data.frame, with any duplicated inputs removed.

Examples

```
## Not run:
transform_method <- function(df) cbind_fast(df, q = df$y * df$y)
data <- data.frame(y = c(1, 2), x = c(0.1, 0.2))
data_transformed <- transform_method(data)
process_transform_throw_error(data, data_transformed, "transform_method")
# transform_method yields data.frame that duplicates input vars - dropping the following
columns: 'y', 'x'
# q
# 1 1
# 2 4

## End(Not run)
```

transform_features *Transform machine learning feature variables*

Description

A function that takes as its arguement another function defining a set of feature variable transformations, and wraps (or adapts) it for use within a machine learning pipeline.

Usage

```
transform_features(.f)
```

Arguments

`.f`	A unary function of a data.frame that returns a new data.frame containing only the transformed feature variables. An error will be thrown if this is not the case.

Value

A unary function of a data.frame that returns the input data.frame with the transformed feature variable columns appended. This function is assigned the classes `"transform_features"` and `"ml_pipeline_section"`.

Examples

```
data <- head(faithful)
f <- transform_features(function(df) {
  data.frame(x1 = (df$waiting - mean(df$waiting)) / sd(df$waiting))
})

f(data)
#    eruptions waiting         x1
#  1     3.600      79  0.8324308
#  2     1.800      54 -1.0885633
#  3     3.333      74  0.4482320
#  4     2.283      62 -0.4738452
#  5     4.533      85  1.2934694
#  6     2.883      55 -1.0117236
```

transform_response *Transform machine learning response variable*

Description

A function that takes as its arguement another function defining a response variable transformation, and wraps (or adapts) it for use within a machine learning pipeline.

Usage

```
transform_response(.f)
```

Arguments

.f	A unary function of a data.frame that returns a new data.frame containing only the transformed response variable. An error will be thrown if this is not the case.

Value

A unary function of a data.frame that returns the input data.frame with the transformed response variable column appended. This function is assigned the classes "transform_response" and "ml_pipeline_section".

Examples

```
data <- head(faithful)
f <- transform_response(function(df) {
  data.frame(y = (df$eruptions - mean(df$eruptions)) / sd(df$eruptions))
})

f(data)
#   eruptions waiting         y
# 1     3.600      79  0.5412808
# 2     1.800      54 -1.3039946
# 3     3.333      74  0.2675649
# 4     2.283      62 -0.8088457
# 5     4.533      85  1.4977485
# 6     2.883      55 -0.1937539
```

---

try_pipeline_func_call

Custom tryCatch configuration for pipeline segment segment functions

---

Description

Custom tryCatch configuration for pipeline segment segment functions

Usage

```
try_pipeline_func_call(.f, arg, func_name)
```

Arguments

.f	Pipleine segment function
arg	Arguement of .f
func_name	(Character string).

Value

Returns the same object as .f does (a data.frame or model object), unless an error is thrown.

Examples

```
## Not run:
data <- data.frame(x = 1:3, y = 1:3 / 10)
f <- function(df) data.frame(p = df$x ^ 2, q = df$wrong)
try_pipeline_func_call(f, data, "f")
# Error in data.frame(p = df$x^2, q = df$wrong) :
#   arguments imply differing number of rows: 3, 0
# --> called from within function: f

## End(Not run)
```

Index