

# Package ‘mlapi’

October 13, 2022

**Type** Package

**Title** Abstract Classes for Building 'scikit-learn' Like API

**Version** 0.1.1

**Author** Dmitriy Selivanov

**Maintainer** Dmitriy Selivanov <selivanov.dmitriy@gmail.com>

**Description** Provides 'R6' abstract classes for building machine learning models with 'scikit-learn' like API. <<https://scikit-learn.org/>> is a popular module for 'Python' programming language which design became de facto a standard in industry for machine learning tasks.

**License** MIT + file LICENSE

**Encoding** UTF-8

**Depends** methods

**Imports** R6 (>= 2.2.1), Matrix (>= 1.1)

**Suggests** knitr

**RoxygenNote** 7.1.1

**VignetteBuilder** knitr

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2022-04-24 13:50:02 UTC

## R topics documented:

fit . . . . .	2
fit_transform . . . . .	2
mlapiDecomposition . . . . .	3
mlapiEstimation . . . . .	5
mlapiEstimationOnline . . . . .	6
mlapiTransformation . . . . .	7
mlapiTransformationOnline . . . . .	8
predict . . . . .	9
transform . . . . .	9

**Index****10**


---

fit	<i>Fits model to data</i>
-----	---------------------------

---

**Description**

Generic function to fit models (inherited from [mlapiEstimation](#))

**Usage**

```
fit(x, model, y = NULL, ...)

## S3 method for class 'Matrix'
fit(x, model, y = NULL, ...)

## S3 method for class 'matrix'
fit(x, model, y = NULL, ...)
```

**Arguments**

x	A matrix like object, should <b>inherit from</b> <code>Matrix</code> <b>or</b> <code>matrix</code>
model	instance of class estimator which should implement method with signature <code>\$fit(x, y, ...)</code>
y	NULL by default. Optional response variable for supervised learning models. Should inherit from vector or <code>Matrix</code> or <code>matrix</code> . See documentation for corresponding models.
...	additional data/model dependent arguments to downstream functions.

**Value**

```
invisible(object$self())
```

---

fit_transform	<i>Fit model to the data, then transforms data</i>
---------------	----------------------------------------------------

---

**Description**

Generic function to fit transformers (inherits from [mlapiTransformation](#))

**Usage**

```
fit_transform(x, model, y = NULL, ...)

## S3 method for class 'Matrix'
fit_transform(x, model, y = NULL, ...)

## S3 method for class 'matrix'
fit_transform(x, model, y = NULL, ...)
```

**Arguments**

x	A matrix like object, should <b>inherit from</b> <code>Matrix</code> <b>or</b> <code>matrix</code>
model	instance of class estimator which should implement method with signature <code>\$fit(x, ...)</code>
y	NULL by default. Optional response variable for supervised models. Should <b>inherit from</b> vector <code>Matrix</code> <b>or</b> <code>matrix</code> . See documentation for corresponding models.
...	additional data/model dependent arguments to downstream functions.

**Value**

Transformed version of the x

---

mlapiDecomposition      *Base abstract class for all decompositions*

---

**Description**

Base class for all **decompositions** which are methods which can decompose matrix into 2 low-dimensional matrices  $x = f(A, B)$ . (Think of this Latent Dirichlet Allocation, Non-negative Matrix Factorization, etc). It inherits from [mlapiTransformation](#) and additionally requires to implement `components` member.

Base class for all **decompositions** which are methods which can decompose matrix into 2 low-dimensional matrices  $x = f(A, B)$  **incrementally**. It inherits from [mlapiDecomposition](#) and additionally requires to implement `partial_fit` method which can learn components incrementally.

**Usage**

```
mlapiDecomposition
```

```
mlapiDecompositionOnline
```

**Format**

R6Class object.

**Fields**

`components` features embeddings. So if matrix is decomposed in a form  $x = f(A, B)$  where  $X = n \times m$ ,  $A = n \times k$ ,  $B = k \times m$  then `B = components`

`components` features embeddings. So if matrix is decomposed in a form  $x = f(A, B)$  where  $X = n \times m$ ,  $A = n \times k$ ,  $B = k \times m$  then `B = components`

## Methods

```

$fit_transform(x, y = NULL, ...)
$transform(x, ...) Performs transformation of the new data (after model was trained)

$fit_transform(x, y = NULL, ...)
$partial_fit(x, y = NULL, ...)
$transform(x, ...) Performs transformation of the new data (after model was trained)

```

## Arguments

**x** A matrix like object, should **inherit from** `Matrix` **or** `matrix`. Allowed classes should be defined in child classes.

**y** `NULL`. Optional target variable. Usually this should be `NULL`. There few cases when it could be used.

**...** additional parameters **with default values**

**x** A matrix like object, should **inherit from** `Matrix` **or** `matrix`. Allowed classes should be defined in child classes.

**y** `NULL`. Optional target variable. Usually this should be `NULL`. There few cases when it could be used.

**...** additional parameters **with default values**

## Examples

```

TruncatedSVD = R6::R6Class(
  classname = "TruncatedSVD",
  inherit = mlapi::mlapiDecomposition,
  public = list(
    initialize = function(rank = 10) {
      private$rank = rank
      super$set_internal_matrix_formats(dense = "matrix", sparse = NULL)
    },
    fit_transform = function(x, ...) {
      x = super$check_convert_input(x)
      private$n_features = ncol(x)
      svd_fit = svd(x, nu = private$rank, nv = private$rank, ...)
      sing_values = svd_fit$d[seq_len(private$rank)]
      result = svd_fit$u %*% diag(x = sqrt(sing_values))
      private$components_ = t(svd_fit$v %*% diag(x = sqrt(sing_values)))
      rm(svd_fit)
      rownames(result) = rownames(x)
      colnames(private$components_) = colnames(x)
      private$fitted = TRUE
      invisible(result)
    },
    transform = function(x, ...) {
      if (private$fitted) {
        stopifnot(ncol(x) == ncol(private$components_))
        lhs = tcrossprod(private$components_)

```

```

        rhs = as.matrix(tcrossprod(private$components_, x))
        t(solve(lhs, rhs))
      }
      else
        stop("Fit the model first with model$fit_transform()!")
    }
  ),
  private = list(
    rank = NULL,
    n_features = NULL,
    fitted = NULL
  )
)
set.seed(1)
model = TruncatedSVD$new(2)
x = matrix(sample(100 * 10, replace = TRUE), ncol = 10)
x_trunc = model$fit_transform(x)
dim(x_trunc)

x_trunc_2 = model$transform(x)
sum(x_trunc_2 - x_trunc)

#' check pipe-compatible S3 interface
x_trunc_2_s3 = transform(x, model)
identical(x_trunc_2, x_trunc_2_s3)

```

---

mlapiEstimation

*Base abstract class for all classification/regression models*


---

## Description

Base class for all estimators. Defines minimal set of members and methods (with signatures) which have to be implemented in child classes.

## Usage

```
mlapiEstimation
```

## Format

R6Class object.

## Methods

```
$fit(x, y, ...)
```

```
$predict(x, ...) Makes predictions on new data (after model was trained)
```

## Arguments

- x** A matrix like object, should **inherit from** `Matrix` or `matrix`. Allowed classes should be defined in child classes.
- y** target - usually vector, but also can be a matrix like object. Allowed classes should be defined in child classes.
- ...** additional parameters **with default values**

## Examples

```
SimpleLinearModel = R6::R6Class(
  classname = "mlapiSimpleLinearModel",
  inherit = mlapi::mlapiEstimation,
  public = list(
    initialize = function(tol = 1e-7) {
      private$tol = tol
      super$set_internal_matrix_formats(dense = "matrix", sparse = NULL)
    },
    fit = function(x, y, ...) {
      x = super$check_convert_input(x)
      stopifnot(is.vector(y))
      stopifnot(is.numeric(y))
      stopifnot(nrow(x) == length(y))

      private$n_features = ncol(x)
      private$coefficients = .lm.fit(x, y, tol = private$tol)[["coefficients"]]
    },
    predict = function(x) {
      stopifnot(ncol(x) == private$n_features)
      x %%% matrix(private$coefficients, ncol = 1)
    }
  ),
  private = list(
    tol = NULL,
    coefficients = NULL,
    n_features = NULL
  )
)
set.seed(1)
model = SimpleLinearModel$new()
x = matrix(sample(100 * 10, replace = TRUE), ncol = 10)
y = sample(c(0, 1), 100, replace = TRUE)
model$fit(as.data.frame(x), y)
res1 = model$predict(x)
# check pipe-compatible S3 interface
res2 = predict(x, model)
identical(res1, res2)
```

---

mlapiEstimationOnline *Base abstract class for all classification/regression models which can be **trained incrementally** (online)*

---

**Description**

Base class for all online estimators. This class inherits from [mlapiEstimation](#) and additionally requires to implement `$partial_fit(x, y, ...)` method. Idea is that user can pass `x`, `y` in chunks and model will be updated/refined incrementally.

**Usage**

```
mlapiEstimationOnline
```

**Format**

R6Class object.

**Methods**

```
$fit(x, y, ...)
```

```
$partial_fit(x, y, ...)
```

```
$predict(x, ...) Makes predictions on new data (after model was trained)
```

**Arguments**

**x** A matrix like object, should **inherit from** `Matrix` or `matrix`. Allowed classes should be defined in child classes.

**y** target - usually vector, but also can be a matrix like object. Allowed classes should be defined in child classes.

**...** additional parameters **with default values**

---

mlapiTransformation *Base abstract class for all transformations*

---

**Description**

Base class for all online transformations.

**Usage**

```
mlapiTransformation
```

**Format**

R6Class object.

**Methods**

```
$fit_transform(x, y = NULL, ...)
```

```
$transform(x, ...) Performs transformation of the new data (after model was trained)
```

**Arguments**

- x** A matrix like object, should **inherit from** `Matrix` **or** `matrix`. Allowed classes should be defined in child classes.
- y** NULL. Optional target variable. Usually this should be NULL. There few cases when it could be used.
- ... additional parameters **with default values**

---

```
mlapiTransformationOnline
```

*Base abstract class for all transformations which can be **trained incrementally** (online)*

---

**Description**

Base class for all online transformations. This class inherits from `mlapiTransformation` and additionally requires to implement `$partial_fit(x, y, ...)` method. Idea is that user can pass `x`, `y` in chunks and model will be updated/refined incrementally.

**Usage**

```
mlapiTransformationOnline
```

**Format**

R6Class object.

**Methods**

```
$fit_transform(x, y = NULL, ...)
```

```
$transform(x, ...) Performs transformation of the new data (after model was trained)
```

**Arguments**

- x** A matrix like object, should **inherit from** `Matrix` **or** `matrix`. Allowed classes should be defined in child classes.
- y** NULL. Optional target variable. Usually this should be NULL. There few cases when it could be used.
- ... additional parameters **with default values**



---

predict	<i>Makes predictions on new data using pre-trained model</i>
---------	--------------------------------------------------------------

---

**Description**

Makes predictions on new data using pre-trained model (inherits from [mlapiEstimation](#))

**Usage**

```
## S3 method for class 'matrix'
predict(object, model, ...)
```

```
## S3 method for class 'Matrix'
predict(object, model, ...)
```

**Arguments**

object	= <b>x</b> in other methods. A matrix like object, should <b>inherit from Matrix or matrix</b>
model	object which <b>inherits</b> class <a href="#">mlapiEstimation</a> which implements method <code>model\$predict(x, ...)</code>
...	additional data/model dependent arguments to downstream functions

---

transform	<i>Transforms new data using pre-trained model</i>
-----------	----------------------------------------------------

---

**Description**

Generic function to transform data with pre-trained model (inherits from [mlapiTransformation](#))

**Usage**

```
## S3 method for class 'Matrix'
transform(`_data`, model, ...)
```

```
## S3 method for class 'matrix'
transform(`_data`, model, ...)
```

**Arguments**

<code>_data</code>	= <b>x</b> in other methods. A matrix like object, should <b>inherit from Matrix or matrix</b>
model	object of class <a href="#">mlapiTransformation</a> which implements method <code>\$transform(x, ...)</code>
...	additional data/model dependent arguments to downstream functions.

# Index

## \* datasets

- mlapiDecomposition, 3
- mlapiEstimation, 5
- mlapiEstimationOnline, 6
- mlapiTransformation, 7
- mlapiTransformationOnline, 8

fit, 2

fit\_transform, 2

mlapiDecomposition, 3, 3

mlapiDecompositionOnline  
(mlapiDecomposition), 3

mlapiEstimation, 2, 5, 7, 9

mlapiEstimationOnline, 6

mlapiTransformation, 2, 3, 7, 8, 9

mlapiTransformationOnline, 8

predict, 9

transform, 9