Package ‘tidyfit’

November 25, 2022

Type Package
Title Regularized Linear Modeling with Tidy Data
Date 2022-11-25
Version 0.6.0
Author Johann Pfitzinger [aut, cre]
Maintainer Johann Pfitzinger <johann.pfitzinger@gmail.com>
Description An extension to the 'R' tidy data environment for automated machine learning. The package allows fitting and cross validation of linear regression and classification algorithms on grouped data.
License GPL-3
Encoding UTF-8
LazyData true
RoxygenNote 7.2.1
URL https://tidyfit.unchartedml.com,
https://github.com/jpfitzinger/tidyfit
Depends R (>= 3.5)
Imports broom, crayon, dials, dplyr, furrr, magrittr, MASS, methods, progressr, purrr, rlang, rsample, stats, tibble, tidyr, utils, yardstick
Suggests arm, bestglm, BMS, CORElearn, e1071, gets, glmnet, hfr, kableExtra, knitr, lme4, lmtest, lubridate, mboost, mRMR, MSwM, pls, quantreg, randomForest, sandwich, shrinkTVP, tidyverse, rmarkdown
VignetteBuilder knitr
NeedsCompilation no
Repository CRAN
Date/Publication 2022-11-25 12:40:05 UTC
R topics documented:

.model.adalasso .................................................. 2
.model.bayes .......................................................... 4
.model.bma ............................................................. 5
.model.boost .......................................................... 7
.model.chisq .......................................................... 8
.model.cor ............................................................ 9
.model.enet ........................................................... 11
.model.gets ........................................................... 12
.model.glm ............................................................ 14
.model.glmm .......................................................... 15
.model.hfr ............................................................ 16
.model.lasso ........................................................... 18
.model.lm ............................................................... 19
.model.mrnr ........................................................... 21
.model.mslm ........................................................... 22
.model.pcr .............................................................. 24
.model.plsr ............................................................. 25
.model.quantile ......................................................... 27
.model.relief .......................................................... 28
.model.rf ............................................................... 30
.model.ridge .......................................................... 31
.model.robust ......................................................... 32
.model.subset .......................................................... 34
.model.svm ............................................................. 35
.model.tvp ............................................................. 37
classify ............................................................... 38
coeff.tidyfit.models .................................................. 40
Factor_Industry_Returns ................................................ 41
fitted.tidyfit.models .................................................. 42
m ................................................................. 43
predict.tidyfit.models ............................................... 45
regress ............................................................... 46
residuals.tidyfit.models ............................................. 48

Index 50

.model.adalasso  Adaptive Lasso regression or classification for tidyfit

Description

Fits an adaptive Lasso regression or classification on a 'tidyFit' R6 class. The function can be used with regress and classify.
Usage

```r
## S3 method for class 'adalasso'
.model(self, data = NULL)
```

Arguments

- `self`: a 'tidyFit' R6 class.
- `data`: a data frame, data frame extension (e.g. a tibble), or a lazy data frame (e.g. from dbplyr or dplyr).

Details

**Hyperparameters:**

- `lambda` (*L1 penalty*)
- `lambda_ridge` (*L2 penalty (default = 0.01) used in the first step to determine the penalty factor*)

**Important method arguments (passed to `m`)**

The adaptive Lasso is a weighted implementation of the Lasso algorithm, with covariate-specific weights obtained using an initial regression fit (in this case, a ridge regression with `lambda = lambda_ridge`, where `lambda_ridge` can be passed as an argument). The adaptive Lasso is computed using the `glmnet::glmnet` function. See `?glmnet` for more details. For classification pass `family = "binomial"` to ... in `m` or use `classify`.

**Implementation**

Features are standardized by default with coefficients transformed to the original scale.

If no hyperparameter grid is passed (`is.null(control$lambda)`), `dials::grid_regular()` is used to determine a sensible default grid. The grid size is 100. Note that the grid selection tools provided by `glmnet::glmnet` cannot be used (e.g. `dfmax`). This is to guarantee identical grids across groups in the tibble.

**Value**

A fitted 'tidyFit' class model.

**Author(s)**

Johann Pfitzinger

**References**


See Also

.model.lasso, .model.enet, .model.ridge and \texttt{m} methods

Examples

```r
# Load data
data <- tidyfit::Factor_Industry_Returns

# Stand-alone function
fit <- m("adalasso", Return ~ ., data, lambda = 0.5)
fit

# Within 'regress' function
fit <- regress(data, Return ~ ., m("adalasso", lambda = c(0.1, 0.5)),
               .mask = c("Date", "Industry"))
coef(fit)
```

---

Bayesian generalized linear regression for tidyfit

Description

Fits a Bayesian regression on a 'tidyFit' R6 class. The function can be used with \texttt{regress} and \texttt{classify}.

Usage

```r
## S3 method for class 'bayes'
.model(self, data = NULL)
```

Arguments

- **self**: a 'tidyFit' R6 class.
- **data**: a data frame, data frame extension (e.g. a tibble), or a lazy data frame (e.g. from \texttt{dbplyr} or \texttt{dplyr}).

Details

Hyperparameters:

None. Cross validation not applicable.

Important method arguments (passed to \texttt{m})

The function provides a wrapper for \texttt{arm::bayesglm}. See \texttt{?bayesglm} for more details.

Implementation

No implementation notes
Value
A fitted 'tidyFit' class model.
A 'tibble'.

Author(s)
Johann Pfitzinger

References

See Also
.model.glm and m methods

Examples
```r
# Load data
data <- tidyfit::Factor_Industry_Returns

# Stand-alone function
fit <- m("bayes", Return ~ ., data)
fit

# Within 'regress' function
fit <- regress(data, Return ~ ., m("bayes"), .mask = c("Date", "Industry"))
coef(fit)
```

Description
Fits a Bayesian model averaging regression on a 'tidyFit' R6 class. The function can be used with regress.

Usage
```r
## S3 method for class 'bma'
.model(self, data = NULL)
```
Arguments

self            a 'tidyFit' R6 class.
data            a data frame, data frame extension (e.g. a tibble), or a lazy data frame (e.g. from dbplyr or dplyr).

Details

Hyperparameters:

None. Cross validation not applicable.

Important method arguments (passed to m)

• iter (number of iteration draws)
• mcmc (model sampler used (default 'bd'))

The function provides a wrapper for BMS::bms. See ?bms for more details.

Implementation

The underlying function automatically generates plotting output, which is not suppressed.

Use coef(fit) to obtain posterior mean, standard deviation as well as posterior inclusion probabilities for the features.

Value

A fitted 'tidyFit' class model.

Author(s)

Johann Pfitzinger

References


See Also

.model.bayes and m methods

Examples

# Load data
data <- tidyfit::Factor_Industry_Returns

# Stand-alone function
fit <- m("bma", Return ~ Mkt-RF + HML + SMB + RMW + CMA, data)
fit

# Within 'regress' function
fit <- regress(data, Return ~ ., m("bma"), .mask = c("Date", "Industry"))
coef(fit)
Description

Fits a gradient boosting regression or classification on a 'tidyFit' R6 class. The function can be used with `regress` and `classify`.

Usage

```r
## S3 method for class 'boost'
.model.boost(self, data = NULL)
```

Arguments

- `self` a 'tidyFit' R6 class.
- `data` a data frame, data frame extension (e.g. a tibble), or a lazy data frame (e.g. from dbplyr or dplyr).

Details

Hyperparameters:
- `mstop` (number of boosting iterations)
- `nu` (step size)

Important method arguments (passed to `m`)

The gradient boosting regression is performed using `mboost::glmboost`. See `?glmboost` for more details.

Implementation

Features are standardized by default with coefficients transformed to the original scale.

If no hyperparameter grid is passed (is.null(control$mstop) and is.null(control$nu)), the default grid is used with `mstop = c(100, 500, 1000, 5000)` and `nu = c(0.01, 0.05, 0.1, 0.15, 0.2, 0.25)`.

Value

- A fitted 'tidyFit' class model.
- A 'tibble'.

Author(s)

Johann Pfitzinger
References


See Also

m method

Examples

# Load data
data <- tidyfit::Factor_Industry_Returns

# Stand-alone function
fit <- m("boost", Return ~ ., data, nu = 0.1, mstop = 100)
fit

# Within 'regress' function
fit <- regress(data, Return ~ ., m("boost", nu = c(0.1, 0.05), mstop = 100),
                 .mask = c("Date", "Industry"))
coef(fit)

Description

Calculates Pearson's Chi-squared test on a 'tidyFit' R6 class. The function can be used with classify.

Usage

## S3 method for class 'chisq'
.chisq(model(self, data = NULL))

Arguments

self a 'tidyFit' R6 class.
data a data frame, data frame extension (e.g. a tibble), or a lazy data frame (e.g. from dbplyr or dtplyr).
.model.cor

Details

**Hyperparameters:**

*None. Cross validation not applicable.*

**Important method arguments (passed to m)**

The function provides a wrapper for `stats::chisq.test`. See `?chisq.test` for more details.

**Implementation**

Results can be viewed using `coef`.

**Value**

A fitted 'tidyFit' class model.

**Author(s)**

Johann Pfitzinger

**See Also**

`.model.cor` and `m` methods

**Examples**

```r
# Load data
data <- tidyfit::Factor_Industry_Returns
data <- dplyr::mutate_at(data, dplyr::vars(-Date, -Industry), dplyr::ntile, n = 10)

# Within 'classify' function
fit <- classify(data, Return ~ ., m("chisq"), .mask = c("Date", "Industry")))
tidyr::unnest(coef(fit), model_info)
```

---

**.model.cor**

*Pearson's correlation for tidyfit*

**Description**

Calculates Pearson’s correlation coefficient on a 'tidyFit' R6 class. The function can be used with `regress`.

**Usage**

```r
## S3 method for class 'cor'
.model(self, data = NULL)
```
Arguments

- **self**: a `tidyFit` R6 class.
- **data**: a data frame, data frame extension (e.g. a tibble), or a lazy data frame (e.g. from dbplyr or dtplyr).

Details

**Hyperparameters:**

None. Cross validation not applicable.

**Important method arguments (passed to m)**

The function provides a wrapper for `stats::cor.test`. See `?cor.test` for more details.

**Implementation**

Results can be viewed using `coef`.

Value

A fitted 'tidyFit' class model.

Author(s)

Johann Pfitzinger

See Also

- `.model.chisq` and `m` methods

Examples

```r
# Load data
data <- tidyfit::Factor_Industry_Returns

# Stand-alone function
fit <- m("cor", Return ~ `Mkt-RF` + HML + SMB, data)
fit

# Within 'regress' function
fit <- regress(data, Return ~ ., m("cor"), .mask = c("Date", "Industry"))
tidyr::unnest(coef(fit), model_info)
```
Description

Fits an ElasticNet regression or classification on a 'tidyFit' R6 class. The function can be used with `regress` and `classify`.

Usage

```r
## S3 method for class 'enet'
.model(self, data = NULL)
```

Arguments

- `self`: a 'tidyFit' R6 class.
- `data`: a data frame, data frame extension (e.g. a tibble), or a lazy data frame (e.g. from dbplyr or dplyr).

Details

Hyperparameters:

- `lambda` (penalty)
- `alpha` (L1-L2 mixing parameter)

Important method arguments (passed to `m`)

The ElasticNet regression is estimated using `glmnet::glmnet`. See `?glmnet` for more details. For classification pass `family = "binomial"` to ... in `m` or use `classify`.

Implementation

If the response variable contains more than 2 classes, a multinomial response is used automatically. An intercept is always included and features are standardized with coefficients transformed to the original scale.

If no hyperparameter grid is passed (is.null(control$lambda) and is.null(control$alpha)), `dials::grid_regular()` is used to determine a sensible default grid. The grid size is 100 for `lambda` and 5 for `alpha`. Note that the grid selection tools provided by `glmnet::glmnet` cannot be used (e.g. `dfmax`). This is to guarantee identical grids across groups in the tibble.

Value

A fitted 'tidyFit' class model.

Author(s)

Johann Pfitzinger
References


See Also

.model.lasso, .model.adalasso, .model.ridge and m methods

Examples

# Load data
data <- tidyfit::Factor_Industry_Returns

# Stand-alone function
fit <- m("enet", Return ~ ., data, lambda = c(0, 0.1), alpha = 0.5)
fit

# Within 'regress' function
fit <- regress(data, Return ~ ., m("enet", alpha = c(0, 0.5), lambda = c(0.1)),
               .mask = c("Date", "Industry"), .cv = "vfold_cv")
coef(fit)

Description

Fits a general-to-specific (GETS) regression on a 'tidyFit' R6 class. The function can be used with regress.

Usage

## S3 method for class 'gets'
.model(self, data = NULL)

Arguments

self a 'tidyFit' R6 class.
data a data frame, data frame extension (e.g. a tibble), or a lazy data frame (e.g. from dbplyr or dtplyr).
Details

Hyperparameters:
None. Cross validation not applicable.

Important method arguments (passed to m)

- `max.paths` *(Number of paths to search)*

The function provides a wrapper for `gets::gets`. See `?gets` for more details.

Implementation
Print output is suppressed by default. Use `print.searchinfo = TRUE` for print output.

Value
A fitted `tidyFit` class model.

Author(s)
Johann Pfitzinger

References

See Also
`.model.robust`, `.model.glm` and `m` methods

Examples

```r
# Load data
data <- tidyfit::Factor_Industry_Returns

# Stand-alone function
fit <- m("gets", Return ~ `Mkt-RF` + HML + SMB, data)
fit

# Within 'regress' function
fit <- regress(data, Return ~ ., m("gets"), .mask = c("Date", "Industry"))
coef(fit)
```
.model.glm Generalized linear regression for tidyfit

Description

Fits a linear or logistic regression on a 'tidyFit' R6 class. The function can be used with `regress` and `classify`.

Usage

```r
## S3 method for class 'glm'
.model(self, data = NULL)
```

Arguments

- `self`: a 'tidyFit' R6 class.
- `data`: a data frame, data frame extension (e.g. a tibble), or a lazy data frame (e.g. from dbplyr or dtplyr).

Details

**Hyperparameters:**

None. Cross validation not applicable.

**Important method arguments (passed to m)**

The function provides a wrapper for `stats::glm`. See `?glm` for more details.

**Implementation**

No implementation notes

Value

A fitted 'tidyFit' class model.

Author(s)

Johann Pfitzinger

See Also

`.model.lm` and `m` methods
Examples

```r
# Load data
data <- tidyfit::Factor_Industry_Returns
data$Return <- ifelse(data$Return > 0, 1, 0)

# Stand-alone function
fit <- m("glm", Return ~ ., data)
fit

# Within 'classify' function
fit <- classify(data, Return ~ ., m("glm"), .mask = c("Date", "Industry"))
coef(fit)
```

### Description

Fits a linear or logistic mixed-effects model (GLMM) on a 'tidyFit' R6 class. The function can be used with `regress` and `classify`.

### Usage

```r
## S3 method for class 'glmm'
.model(self, data = NULL)
```

### Arguments

- `self`: a 'tidyFit' R6 class.
- `data`: a data frame, data frame extension (e.g. a tibble), or a lazy data frame (e.g. from `dbplyr` or `dtplyr`).

### Details

**Hyperparameters:**

*None. Cross validation not applicable.*

**Important method arguments (passed to m)**

The function provides a wrapper for `lme4::glmer`. See `?glmer` for more details.

**Implementation**

*No implementation notes*

### Value

A fitted 'tidyFit' class model.
Hierarchical feature regression for tidyfit

Description

Fits a hierarchical feature regression on a 'tidyFit' R6 class. The function can be used with `regress`.

Usage

```r
## S3 method for class 'hfr'
.model(self, data = NULL)
```

Arguments

- `self`: a 'tidyFit' R6 class.
- `data`: a data frame, data frame extension (e.g. a tibble), or a lazy data frame (e.g. from `dbplyr` or `dtplyr`).
Details

**Hyperparameters:**

- kappa_grid *(proportional size of regression graph)*

**Important method arguments (passed to m)**

The hierarchical feature regression is estimated using the `hfr::cv.hfr` function. See `?cv.hfr` for more details.

**Implementation**

Features are standardized by default with coefficients transformed to the original scale.

If no hyperparameter grid is provided (`is.null(control$kappa_grid)`), the default is `seq(0, 1, by = 0.1)`.

**Value**

A fitted `tidyFit` class model.

**Author(s)**

Johann Pfitzinger

**References**


**See Also**

`.model.plsr` and `m` methods

**Examples**

```r
# Load data
data <- tidyfit::Factor_Industry_Returns

# Stand-alone function
fit <- m("hfr", Return ~ ., data, kappa_grid = 0.5)
fit

# Within 'regress' function
fit <- regress(data, Return ~ ., m("hfr", kappa_grid = c(0.1, 0.5)),
               .mask = c("Date", "Industry"))
coef(fit)
```
Description

Fits a linear regression or classification with L1 penalty on a 'tidyFit' R6 class. The function can be used with `regress` and `classify`.

Usage

```r
## S3 method for class 'lasso'
.model(self, data = NULL)
```

Arguments

- `self`: a 'tidyFit' R6 class.
- `data`: a data frame, data frame extension (e.g. a tibble), or a lazy data frame (e.g. from dbplyr or dtplyr).

Details

**Hyperparameters:**

- `lambda` (*L1 penalty*)

**Important method arguments (passed to m)**

The Lasso regression is estimated using `glmnet::glmnet` with `alpha = 1`. See `?glmnet` for more details. For classification pass `family = "binomial"` to ... in `m` or use `classify`.

**Implementation**

If the response variable contains more than 2 classes, a multinomial response is used automatically.

Features are standardized by default with coefficients transformed to the original scale.

If no hyperparameter grid is passed (is.null(control$lambda), `dials::grid_regular()` is used to determine a sensible default grid. The grid size is 100. Note that the grid selection tools provided by `glmnet::glmnet` cannot be used (e.g. `dfmax`). This is to guarantee identical grids across groups in the tibble.

**Value**

A fitted 'tidyFit' class model.

**Author(s)**

Johann Pfitzinger
References


See Also

.model.enet, .model.ridge, .model.adalasso and m methods

Examples

# Load data
data <- tidyfit::Factor_Industry_Returns

# Stand-alone function
fit <- m("lasso", Return ~ ., data, lambda = 0.5)
fit

# Within 'regress' function
fit <- regress(data, Return ~ ., m("lasso", lambda = c(0.1, 0.5)),
               .mask = c("Date", "Industry"))
coef(fit)

.model.lm

Linear regression for tidyfit

Description

Fits a linear regression on a 'tidyFit' R6 class. The function can be used with regress.

Usage

## S3 method for class 'lm'
[model(self, data = NULL)

Arguments

default

data

a 'tidyFit' R6 class.
a data frame, data frame extension (e.g. a tibble), or a lazy data frame (e.g. from dbplyr or dtplyr).
Details

Hyperparameters:

None. Cross validation not applicable.

Important method arguments (passed to \texttt{m})

The function provides a wrapper for \texttt{stats::lm}. See \texttt{?lm} for more details.

Implementation

An argument \texttt{vcov.} can be passed in control or to \ldots{} in \texttt{m} to estimate the model with robust standard errors. \texttt{vcov.} can be one of "BS", "HAC", "HC" and "OPG" and is passed to the \texttt{sandwich} package.

Value

A fitted 'tidyFit' class model.

Author(s)

Johann Pfitzinger

See Also

\texttt{.model.robust}, \texttt{.model.glm} and \texttt{m} methods

Examples

# Load data
data <- tidyfit::Factor_Industry_Returns

# Stand-alone function
fit <- m("lm", Return ~ `Mkt-RF` + HML + SMB, data)
fit

# Within 'regress' function
fit <- regress(data, Return ~ ., m("lm"), .mask = c("Date", "Industry"))
coef(fit)

# With robust standard errors
fit <- m("lm", Return ~ `Mkt-RF` + HML + SMB, data, vcov. = "HAC")
fit
Minimum redundancy, maximum relevance feature selection for tidyfit

Description

Selects features for continuous or (ordered) factor data using MRMR on a `tidyFit` R6 class. The function can be used with `regress` and `classify`.

Usage

```r
## S3 method for class 'mrMr'
.model(self, data = NULL)
```

Arguments

- `self` a `tidyFit` R6 class.
- `data` a data frame, data frame extension (e.g. a tibble), or a lazy data frame (e.g. from dbplyr or dtplyr).

Details

Hyperparameters:

None. Cross validation not applicable.

Important method arguments (passed to `m`)

- `feature_count` (number of features to select)
- `solution_count` (ensemble size)

The MRMR algorithm is estimated using the mRMR::mRMR.ensemble function. See `?mRMR.ensemble` for more details.

Implementation

Use with `regress` for regression problems and with `classify` for classification problems. The selected features can be obtained using `coef`.

The MRMR objects have no `predict` and related methods.

Value

A fitted `tidyFit` class model.

Author(s)

Johann Pfitzinger
References


See Also

\texttt{m} methods

Examples

# Load data
data <- tidyfit::Factor_Industry_Returns

# Stand-alone function
fit <- m("mrmr", Return ~ ., data, feature_count = 3)
coef(fit)

# Within 'regress' function
fit <- regress(data, Return ~ ., m("mrmr", feature_count = 3),
               .mask = c("Date", "Industry"))
coef(fit)

Description

Fits a Markov-Switching regression on a 'tidyFit' R6 class. The function can be used with \texttt{regress}.

Usage

## S3 method for class 'mslm'
$model(self, data = NULL)

Arguments

self a 'tidyFit' R6 class.
data a data frame, data frame extension (e.g. a tibble), or a lazy data frame (e.g. from dbplyr or dtplyr).

Details

Hyperparameters:

\textit{None. Cross validation not applicable.}

Important method arguments (passed to \texttt{m})

- \texttt{k} (the number of regimes)
The function provides a wrapper for `MSwM::msmFit`. See `?msmFit` for more details.

**Implementation**

Note that only the regression method with 'lm' is implemented at this stage.

An argument `index_col` can be passed, which allows a custom index to be added to `coef(m("mslm"))` (e.g. a date index).

If no `sw` argument is passed, all coefficients are permitted to switch between regimes.

**Value**

A fitted 'tidyFit' class model.

**Author(s)**

Johann Pfitzinger

**References**


**See Also**

`.model.tvp` and `m` methods

**Examples**

```r
# Load data
data <- tidyfit::Factor_Industry_Returns
data <- dplyr::filter(data, Industry == "HiTec", Date >= 201801)
data <- dplyr::select(data, -Industry)

ctr <- list(maxiter = 100, parallelization = FALSE)

# Stand-alone function
fit <- m("mslm", Return ~ HML, data, index_col = "Date", k = 2, control = ctr)
fit

# Within 'regress' function
fit <- regress(data, Return ~ HML,
               m("mslm", index_col = "Date", k = 2, control = ctr))
tidyr::unnest(coef(fit), model_info)
```
Principal Components Regression for tidyfit

Description

Fits a principal components regression on a ‘tidyFit’ R6 class. The function can be used with `regress`.

Usage

```r
## S3 method for class 'pcr'
.model(self, data = NULL)
```

Arguments

- `self` a 'tidyFit' R6 class.
- `data` a data frame, data frame extension (e.g. a tibble), or a lazy data frame (e.g. from dbplyr or dtplyr).

Details

Hyperparameters:

- `ncomp` (number of components)
- `ncomp_pct` (number of components, percentage of features)

Important method arguments (passed to `m`)

The principal components regression is fitted using `pls` package. See `?pcr` for more details.

Implementation

Covariates are standardized, with coefficients back-transformed to the original scale. An intercept is always included.

If no hyperparameter grid is passed (is.null(control$ncomp) & is.null(control$ncomp_pct)), the default is `ncomp_pct = seq(0, 1, length.out = 20)`, where 0 results in one component and 1 results in the number of features.

When 'jackknife = TRUE' is passed (and a 'validation' method is chosen), `coef` also returns the jack-knife standard errors, t-statistics and p-values.

Note that at present `pls` does not offer weighted implementations or non-gaussian response. The method can therefore only be used with `regress`.

Value

A fitted 'tidyFit' class model.

Author(s)

Johann Pfitzinger
References


See Also

.model.plsr and m methods

Examples

# Load data
data <- tidyfit::Factor_Industry_Returns
data <- dplyr::filter(data, Industry == "HiTec")
data <- dplyr::select(data, -Industry)

# Stand-alone function
fit <- m("pcr", Return ~ ., data, ncomp = 1:3)
fit

# Within 'regress' function
fit <- regress(data, Return ~ .,  
m("pcr", jackknife = TRUE, validation = "LOO", ncomp_pct = 0.5),  
    .mask = c("Date"))
tidyR::unnest(coef(fit), model_info)
Details

**Hyperparameters:**

- `ncomp` (*number of components*)
- `ncomp_pct` (*number of components, percentage of features*)

**Important method arguments (passed to `m`)**

The partial least squares regression is fitted using `pls` package. See `?plsr` for more details.

**Implementation**

Covariates are standardized, with coefficients back-transformed to the original scale. An intercept is always included.

If no hyperparameter grid is passed (`is.null(control$ncomp) & is.null(control$ncomp_pct)`), the default is `ncomp_pct = seq(0, 1, length.out = 20)`, where 0 results in one component and 1 results in the number of features.

When 'jackknife = TRUE' is passed (and a 'validation' method is chosen), `coef` also returns the jack-knife standard errors, t-statistics and p-values.

Note that at present `pls` does not offer weighted implementations or non-gaussian response. The method can therefore only be used with `regress`.

**Value**

A fitted 'tidyFit' class model.

**Author(s)**

Johann Pfitzinger

**References**


**See Also**

`.model.pcr` and `m` methods

**Examples**

```r
# Load data
data <- tidyfit::Factor_Industry_Returns
data <- dplyr::filter(data, Industry == "HiTec")
data <- dplyr::select(data, -Industry)

# Stand-alone function
fit <- m("plsr", Return ~ ., data, ncomp = 1:3)
fit

# Within 'regress' function
```
fit <- regress(data, Return ~ ..,
  m("pcr", jackknife = TRUE, validation = "LOO", ncomp_pct = 0.5),
  .mask = c("Date"))
tidyr::unnest(coef(fit), model_info)

---

**Quantile regression for tidyfit**

**Description**

Fits a linear quantile regression on a 'tidyFit' R6 class. The function can be used with `regress`.

**Usage**

```r
## S3 method for class 'quantile'
.model(self, data = NULL)
```

**Arguments**

- `self`: a 'tidyFit' R6 class.
- `data`: a data frame, data frame extension (e.g. a tibble), or a lazy data frame (e.g. from dbplyr or dplyr).

**Details**

- **Hyperparameters:**
  - None. Cross validation not applicable.

- **Important method arguments (passed to `m`)**
  - `tau` (the quantile(s) to be estimated)

  The function provides a wrapper for `quantreg::rq`. See `?rq` for more details. The argument `tau` is the chosen quantile (default `tau = 0.5`).

**Implementation**

No implementation notes

**Value**

A fitted 'tidyFit' class model.

**Author(s)**

Johann Pfitzinger
References

See Also
.model.lm, .model.bayes and m methods

Examples
```r
# Load data
data <- tidyfit::Factor_Industry_Returns

# Stand-alone function
fit <- m("quantile", Return ~ ., data, tau = 0.5)
fit

# Within 'regress' function
fit <- regress(data, Return ~ .,
               m("quantile", tau = c(0.1, 0.5, 0.9)),
               .mask = c("Date", "Industry"))
coef(fit)
```

Description
Selects features for continuous or factor data using ReliefF on a 'tidyFit' R6 class. The function can be used with regress and classify.

Usage
```r
## S3 method for class 'relief'
.model(self, data = NULL)
```

Arguments
- `self` - a 'tidyFit' R6 class.
- `data` - a data frame, data frame extension (e.g. a tibble), or a lazy data frame (e.g. from dbplyr or dtplyr).
Details

Hyperparameters:
None. Cross validation not applicable.

Important method arguments (passed to m)

• estimator (selection algorithm to use (default is 'ReliefFqualK'))

The Relief algorithm is estimated using the CORElearn::attrEval function. See ?attrEval for more details.

Implementation

Use with regress for regression problems and with classify for classification problems. coef returns the score for each feature. Select the required number of features with the largest scores.

The Relief objects have no predict and related methods.

Value

A fitted 'tidyFit' class model.

Author(s)

Johann Pfitzinger

References


See Also

.model.mrmr and m methods

Examples

# Load data
data <- tidyfit::Factor_Industry_Returns
data <- dplyr::filter(data, Industry == "HiTec")
data <- dplyr::select(data, -Date, -Industry)

# Stand-alone function
fit <- m("relief", Return ~ ., data)
coef(fit)

# Within 'regress' function
fit <- regress(data, Return ~ ., m("relief"))
coef(fit)
Description

Fits a random forest on a 'tidyFit' R6 class. The function can be used with `regress` and `classify`.

Usage

```r
## S3 method for class 'rf'
.model(self, data = NULL)
```

Arguments

- `self`: a 'tidyFit' R6 class.
- `data`: a data frame, data frame extension (e.g. a tibble), or a lazy data frame (e.g. from dbplyr or dplyr).

Details

Hyperparameters:

- `ntree` (number of trees)
- `mtry` (number of variables randomly sampled at each split)

Important method arguments (passed to `m`)

The function provides a wrapper for `randomForest::randomForest`. See `?randomForest` for more details.

Implementation

The random forest is always fit with `importance = TRUE`. The feature importance values are extracted using `coef()`.

Value

A fitted 'tidyFit' class model.

Author(s)

Johann Pfitzinger

References


See Also

`.model.svm`, `.model.boost` and `m` methods
Examples

```r
# Load data
data <- tidyfit::Factor_Industry_Returns
data <- dplyr::filter(data, Industry == "HiTec")
data <- dplyr::select(data, -Date, -Industry)

# Stand-alone function
fit <- m("rf", Return ~ ., data)
fit

# Within 'regress' function
fit <- regress(data, Return ~ ., m("rf"))
tidyr::unnest(coef(fit), model_info)
```

Description

Fits a linear regression or classification with L2 penalty on a 'tidyFit' R6 class. The function can be used with `regress` and `classify`.

Usage

```r
## S3 method for class 'ridge'
.model(self, data = NULL)
```

Arguments

- `self` a tidyFit R6 class.
- `data` a data frame, data frame extension (e.g. a tibble), or a lazy data frame (e.g. from dbplyr or dtplyr).

Details

Hyperparameters:

- `lambda` (L2 penalty)

Important method arguments (passed to `m`)

The ridge regression is estimated using `glmnet::glmnet` with `alpha = 0`. See `?glmnet` for more details. For classification pass `family = "binomial"` to ... in `m` or use `classify`.

Implementation

If the response variable contains more than 2 classes, a multinomial response is used automatically. Features are standardized by default with coefficients transformed to the original scale.
If no hyperparameter grid is passed (is.null(control$lambda)), dials::grid_regular() is used to determine a sensible default grid. The grid size is 100. Note that the grid selection tools provided by glmnet::glmnet cannot be used (e.g. dfmax). This is to guarantee identical grids across groups in the tibble.

Value
A fitted tidyFit class model.

Author(s)
Johann Pfitzinger

References

See Also
$model.lasso, .model.adalasso, .model.enet and m methods

Examples

# Load data
data <- tidyfit::Factor_Industry_Returns

# Stand-alone function
fit <- m("ridge", Return ~ ., data, lambda = 0.5)
fit

# Within 'regress' function
fit <- regress(data, Return ~ ., m("ridge", lambda = c(0.1, 0.5)),
               .mask = c("Date", "Industry"))
coef(fit)

---

$model.robust Robust regression for tidyfit

Description
Fits a robust linear regression on a 'tidyFit' R6 class. The function can be used with regress.

Usage

## S3 method for class 'robust'
$model(self, data = NULL)
Arguments

- **self**: a 'tidyFit' R6 class.
- **data**: a data frame, data frame extension (e.g. a tibble), or a lazy data frame (e.g. from dbplyr or dtplyr).

Details

**Hyperparameters:**

*None. Cross validation not applicable.*

**Important method arguments (passed to m)**

- `method` (estimation algorithm, e.g. 'M', 'MM')

The function provides a wrapper for `MASS::rlm`. See `?rlm` for more details.

**Implementation**

An argument `vcov.` can be passed in control or to `...` in `m` to estimate the model with robust standard errors. `vcov.` can be one of "BS", "HAC", "HC" and "OPG" and is passed to the `sandwich` package.

Value

A fitted 'tidyFit' class model.

Author(s)

Johann Pfitzinger

References


See Also

`model.lm` and `m` methods

Examples

```r
# Load data
data <- tidyfit::Factor_Industry_Returns

# Stand-alone function
fit <- m("robust", Return ~ `Mkt-RF` + HML + SMB, data)
fit

# Within 'regress' function
fit <- regress(data, Return ~ ., m("robust"), .mask = c("Date", "Industry"))
coef(fit)
```
# With robust standard errors
fit <- m("robust", Return ~ `Mkt-RF` + HML + SMB, data, vcov. = "HAC")
tidyr::unnest(coef(fit), model_info)

---

**Best subset regression and classification for tidyfit**

**Description**
Fits a best subset regression or classification on a 'tidyFit' R6 class. The function can be used with `regress` and `classify`.

**Usage**
```r
## S3 method for class 'subset'
.model(self, data = NULL)
```

**Arguments**
- `self` a 'tidyFit' R6 class.
- `data` a data frame, data frame extension (e.g. a tibble), or a lazy data frame (e.g. from `dbplyr` or `dtplyr`).

**Details**

**Hyperparameters:**
*None. Cross validation not applicable.*

**Important method arguments (passed to `m`)**
- `method` (e.g. 'forward', 'backward')
- `IC` (information criterion, e.g. 'AIC')

The best subset regression is estimated using `bestglm::bestglm` which is a wrapper around `leaps::regsubsets` for the regression case, and performs an exhaustive search for the classification case. See `?bestglm` for more details.

**Implementation**
Forward or backward selection can be performed by passing `method = "forward"` or `method = "backward"` to `m`.

**Value**
A fitted 'tidyFit' class model.

**Author(s)**
Johann Pfitzinger
References


See Also

`.model.lm` and `m` methods

Examples

```r
# Load data
data <- tidyfit::Factor_Industry_Returns

# Stand-alone function
fit <- m("subset", Return ~ ., data, method = c("forward", "backward"))
tidy::unnest(fit, settings)

# Within 'regress' function
fit <- regress(data, Return ~ ., m("subset", method = "forward"),
               .mask = c("Date", "Industry"))
coef(fit)
```

---

**.model.svm**

Support vector regression or classification for tidyfit

**Description**

Fits a support vector regression or classification on a 'tidyFit' R6 class. The function can be used with `regress` or `classify`.

**Usage**

```r
## S3 method for class 'svm'
.model(self, data = NULL)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>self</code></td>
<td>a 'tidyFit' R6 class.</td>
</tr>
<tr>
<td><code>data</code></td>
<td>a data frame, data frame extension (e.g. a tibble), or a lazy data frame (e.g. from dbplyr or dtplyr).</td>
</tr>
</tbody>
</table>
Details

Hyperparameters:

- cost (cost of constraint violation)
- epsilon (epsilon in the insensitive-loss function)

Important method arguments (passed to m)

The function provides a wrapper for e1071::svm. See ?svm for more details.

Implementation

The default value for the kernel argument is set to 'linear'. If set to a different value, no coefficients will be returned.

Value

A fitted 'tidyFit' class model.

Author(s)

Johann Pfitzinger

References


See Also

.model.boost, .model.lasso and m methods

Examples

# Load data
data <- tidyfit::Factor_Industry_Returns
data <- dplyr::filter(data, Industry == "HiTec")

# Stand-alone function
fit <- m("svm", Return ~ `Mkt-RF` + HML + SMB, data, cost = 0.1)
fit

# Within 'regress' function
fit <- regress(data, Return ~ ., m("svm", cost = 0.1),
               .mask = c("Date", "Industry"))
coef(fit)
Bayesian Time-Varying Regression for tidyfit

Description

Fits a Bayesian time-varying regression on a 'tidyFit' R6 class. The function can be used with `regress`.

Usage

```r
## S3 method for class 'tvp'
.model(self, data = NULL)
```

Arguments

- `self`: a 'tidyFit' R6 class.
- `data`: a data frame, data frame extension (e.g. a tibble), or a lazy data frame (e.g. from dbplyr or dplyr).

Details

- **Hyperparameters:**
  None. Cross validation not applicable.

- **Important method arguments (passed to m)**
  - `mod_type`
  - `niter` (number of MCMC iterations)

The function provides a wrapper for `shrinkTVP::shrinkTVP`. See `?shrinkTVP` for more details.

Implementation

An argument `index_col` can be passed, which allows a custom index to be added to `coef(m("tvp"))` (e.g. a date index, see Examples).

Value

A fitted 'tidyFit' class model.

Author(s)

Johann Pfitzinger

References

### classify

**Classification on tidy data**

#### Description

This function is a wrapper to fit many different types of linear classification models on a (grouped) tibble.

#### Arguments

- `data`: a data frame, data frame extension (e.g. a tibble), or a lazy data frame (e.g. from dbplyr or dplyr). The data frame can be grouped.
- `formula`: an object of class "formula": a symbolic description of the model to be fitted.
- `...`: name-function pairs of models to be estimated. See 'Details'.
- `.cv`: type of 'rsample' cross validation procedure to use to determine optimal hyperparameter values. Default is `.cv = "none"`. See 'Details'.
- `.cv_args`: additional settings to pass to the 'rsample' cross validation function.
- `.weights`: optional name of column containing sample weights.
- `.mask`: optional vector of columns names to ignore. Can be useful when using 'y ~ .' formula syntax.
- `.return_slices`: logical. Should the output of individual cross validation slices be returned or only the final fit. Default is `.return_slices=FALSE`.
- `.tune_each_group`: logical. Should optimal hyperparameters be selected for each group or once across all groups. Default is `.tune_each_group=TRUE`.
- `.force_cv`: logical. Should models be evaluated across all cross validation slices, even if no hyperparameters are tuned. Default is `.force_cv=TRUE`.

#### Examples

```r
# Load data
data <- tidyfit::Factor_Industry_Returns
data <- dplyr::filter(data, Industry == "HiTec")
data <- dplyr::select(data, -Industry)

# Stand-alone function (using low niter for illustration)
fit <- m("tvp", Return ~ ., data, index_col = "Date", niter = 50)
fit

# Within 'regress' function (using low niter for illustration)
fit <- regress(data, Return ~ ., m("tvp", niter = 50, index_col = "Date"))
tidyr::unnest(coef(fit), model_info)
```

#### See Also

'.model.bayes', '.model.mslm' and `m` methods
classify

Details
classify fits all models passed in ... using the m function. The models can be passed as name-
function pairs (e.g. ols = m("lm")) or without including a name.

Hyperparameters are tuned automatically using the ‘.cv’ and ‘.cv_args’ arguments, or can be passed
to m() (e.g. lasso = m("lasso", lambda = 0.5)). See the individual model functions (?m()) for
an overview of hyperparameters.

Cross validation is performed using the ‘rsample’ package with possible methods including

- ‘initial_split’ (simple train-test split)
- ‘initial_time_split’ (train-test split with retained order)
- ‘vfold_cv’ (aka kfold cross validation)
- ‘loo_cv’ (leave-one-out)
- ‘rolling_origin’ (generalized time series cross validation, e.g. rolling or expanding windows)
- ‘sliding_window’, ‘sliding_index’, ‘sliding_period’ (specialized time series splits)
- ‘bootstraps’
- ‘group_vfold_cv’, ‘group_bootstraps’

See package documentation for ‘rsample’ for all available methods.

The negative log loss is used to validate performance in the cross validation.

Note that arguments for weights are automatically passed to the functions by setting the ‘.weights’
argument. Weights are also considered during cross validation by calculating weighted versions of
the cross validation loss function.

classify can handle both binomial and multinomial response distributions, however not all under-
lying methods are capable of handling a multinomial response.

Value

A tidyfit.models frame containing model details for each group.

The 'tidyfit.models' frame consists of 4 different components:

1. A group of identifying columns (e.g. model name, data groups, grid IDs)
2. A 'model_object' column, which contains the fitted model.
3. A nested 'settings' column containing model arguments and hyperparameters
4. Columns showing errors, warnings and messages (if applicable)

Coefficients, predictions, fitted values or residuals can be accessed using the built-in coef, predict,
fitted and resid methods. Note that all coefficients are transformed to ensure comparability
across methods.

Author(s)

Johann Pfitzinger

See Also

regress, coef.tidyfit.models and predict.tidyfit.models method
Examples

data <- tidyfit::Factor_Industry_Returns
data <- dplyr::mutate(data, Return = ifelse(Return > 0, 1, 0))
fit <- classify(data, Return ~ ., m("lasso", lambda = c(0.001, 0.1)), .mask = c("Date", "Industry"))

# Print the models frame
tidyr::unnest(fit, settings)

# View coefficients
coef(fit)

---

tidyfit.models  Extract coefficients from a tidyfit.models frame

Description

The function extracts and prepares coefficients from all models in a tidyfit.models frame and outputs a tidy frame of estimates.

Usage

```r
## S3 method for class 'tidyfit.models'

coef(
  object,
  ..., .add_bootstrap_interval = FALSE,
  .bootstrap_alpha = 0.05,
  .keep_grid_id = FALSE
)
```

Arguments

- `object` model.frame created using `regress, classify` or `m`
- `...` currently not used
- `.add_bootstrap_interval` calculate bootstrap intervals for the parameters. See 'Details'.
- `.bootstrap_alpha` confidence level used for the bootstrap interval. Default is `.bootstrap_alpha = 0.05`
- `.keep_grid_id` boolean. By default the grid ID column is dropped, if there is only one unique setting per model or group. `.keep_grid_id = TRUE` ensures that the column is never dropped.
Details

The function uses the 'model_object' column in a tidyfit.model frame to return a data frame of estimated coefficients.

Results are 'tidied' using broom::tidy whenever possible.

All coefficients are transformed to ensure statistical comparability. For instance, standardized coefficients are always transformed back to the original data scale, naming conventions are harmonized etc.

**Bootstrap intervals:**

Bootstrap intervals can be calculated using rsample::int_pctl. Only set .add_bootstrap_interval = TRUE if you are using .cv = "bootstraps" in combination with .return_slices = TRUE to generate the model frame.

Value

A 'tibble'.

Author(s)

Johann Pfitzinger

See Also

predict.tidyfit.models, fitted.tidyfit.models and residuals.tidyfit.models

Examples

data <- tidyfit::Factor_Industry_Returns
fit <- regress(data, Return ~ ., m("lm"), .mask = c("Date", "Industry"))
coef(fit)

References

https://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html
fitted.tidyfit.models  Obtain fitted values from models in a tidyfit.models frame

Description

The function generates fitted values for all models in a tidyfit.models frame and outputs a tidy frame.

Usage

## S3 method for class 'tidyfit.models'
fitted(object, ...)

Arguments

object  model.frame created using regress, classify or m

...  currently not used

Details

The function uses the 'model_object' column in a tidyfit.model frame to return fitted values for each model.

Value

A 'tibble'.

Author(s)

Johann Pfitzinger

See Also

coef.tidyfit.models, predict.tidyfit.models and residuals.tidyfit.models

Examples

data <- dplyr::group_by(tidyfit::Factor_Industry_Returns, Industry)
fit <- regress(data, Return ~ ., m("lm"), .mask = "Date")
fitted(fit)
Generic model wrapper for tidyfit

Description

The function can fit various regression or classification models and returns the results as a tibble. 
m() can be used in conjunction with regress and classify, or as a stand-alone function.

Usage

m(model_method, formula = NULL, data = NULL, ...)

Arguments

model_method The name of the method to fit. See Details.
formula an object of class "formula": a symbolic description of the model to be fitted.
data a data frame, data frame extension (e.g. a tibble), or a lazy data frame (e.g. from dbplyr or dtplyr).
... Additional arguments passed to the underlying method function (e.g. lm or glm).

Details

model_method specifies the model to fit to the data and can take one of several options:

**Linear (generalized) regression or classification:**
"lm" performs an OLS regression using stats::lm. See .model.lm for details.
"glm" performs a generalized regression or classification using stats::glm. See .model.glm for details.
"robust" performs a robust regression using MASS::rlm. See .model.robust for details.
"quantile" performs a quantile regression using quantreg::rq. See .model.quantile for details.

**Regression and classification with L1 and L2 penalties:**
"lasso" performs a linear regression or classification with L1 penalty using glmnet::glmnet. See .model.lasso for details.
"ridge" performs a linear regression or classification with L2 penalty using glmnet::glmnet. See .model.ridge for details.
"adalasso" performs an Adaptive Lasso regression or classification using glmnet::glmnet. See .model.adalasso for details.
"enet" performs a linear regression or classification with L1 and L2 penalties using glmnet::glmnet. See .model.enet for details.

**Other Machine Learning:**
"boost" performs gradient boosting regression or classification using mboost::glmboost. See .model.boost for details.
"rf" performs a random forest regression or classification using `randomForest::randomForest`. See `.model.rf` for details.

"svm" performs a support vector regression or classification using `e1071::svm`. See `.model.svm` for details.

**Factor regressions:**
"pcr" performs a principal components regression using `pls::pcr`. See `.model.pcr` for details.
"plsr" performs a partial least squares regression using `pls::plsr`. See `.model.plsr` for details.
"hfr" performs a hierarchical feature regression using `hfr::hfr`. See `.model.hfr` for details.

**Best subset selection:**
"subset" performs a best subset regression or classification using `bestglm::bestglm` (wrapper for `leaps`). See `.model.subset` for details.
"gets" performs a general-to-specific regression using `gets::gets`. See `.model.gets` for details.

**Bayesian methods:**
"bayes" performs a Bayesian generalized regression or classification using `arm::bayesglm`. See `.model.bayes` for details.
"bma" performs a Bayesian model averaging regression using `BMS::bms`. See `.model.bma` for details.
"tvp" performs a Bayesian time-varying parameter regression using `shrinkTVP::shrinkTVP`. See `.model.tvp` for details.

**Mixed-effects modeling:**
"glmm" performs a mixed-effects GLM using `lme4::glmer`. See `.model.glmm` for details.

**Specialized time series methods:**
"mslm" performs a Markov-switching regression using `MSwM::msmFit`. See `.model.mslm` for details.

**Feature selection:**
"cor" calculates Pearson’s correlation coefficient using `stats::cor.test`. See `.model.cor` for details.
"chisq" calculates Pearson’s Chi-squared test using `stats::chisq.test`. See `.model.chisq` for details.
"mrmr" performs a minimum redundancy, maximum relevance features selection routine using `mRMR::mRMR.ensemble`. See `.model.mrmr` for details.
"relief" performs a ReliefF feature selection routine using `CORElearn::attrEval`. See `.model.relief` for details.

When called without `formula` and `data` arguments, the function returns a ‘`tidyfit.models`’ data frame with unfitted models.

**Value**
A ‘`tidyfit.models`’ data frame.
Author(s)

Johann Pfitzinger

See Also

regress and classify methods

Examples

# Load data
data <- tidyfit::Factor_Industry_Returns

# Stand-alone function
fit <- m("lm", Return ~ ., data)
f

# Within 'regress' function
fit <- regress(data, Return ~ ., m("lm"), .mask = "Date")
f

predict.tidyfit.models

Predict using a tidyfit.models frame

Description

The function generates predictions for all models in a tidyfit.models frame and outputs a tidy frame.

Usage

## S3 method for class 'tidyfit.models'
predict(object, newdata, ..., .keep_grid_id = FALSE)

Arguments

object  model.frame created using regress, classify or m
newdata New values at which predictions are to made
... currently not used
.keep_grid_id boolean. By default the grid ID column is dropped, if there is only one unique setting per model or group. .keep_grid_id = TRUE ensures that the column is never dropped.
Details

The function uses the ‘model_object’ column in a tidyfit.model frame to return predictions using the newdata argument for each model.

When the response variable is found in newdata, it is automatically included as a ‘truth’ column.

Value

A ‘tibble’.

Author(s)

Johann Pfitzinger

See Also

coef.tidyfit.models, residuals.tidyfit.models and fitted.tidyfit.models

Examples

data <- dplyr::group_by(tidyfit::Factor_Industry_Returns, Industry)
fit <- regress(data, Return ~ ., m("lm"), .mask = "Date")
predict(fit, data)

---

regress Linear regression on tidy data

Description

This function is a wrapper to fit many different types of linear regression models on a (grouped) tibble.

Arguments

- `.data` a data frame, data frame extension (e.g. a tibble), or a lazy data frame (e.g. from dbplyr or dtplyr). The data frame can be grouped.
- `formula` an object of class "formula": a symbolic description of the model to be fitted.
- `...` name-function pairs of models to be estimated. See ‘Details’.
- `.cv` type of ‘rsample’ cross validation procedure to use to determine optimal hyper-parameter values. Default is `.cv = "none"`. See ‘Details’.
- `.cv_args` additional settings to pass to the ‘rsample’ cross validation function.
- `.weights` optional name of column containing sample weights.
- `.mask` optional vector of columns names to ignore. Can be useful when using ‘y ~ .’ formula syntax.
.return_slices logical. Should the output of individual cross validation slices be returned or only the final fit. Default is .return_slices=FALSE.

tune_each_group logical. Should optimal hyperparameters be selected for each group or once across all groups. Default is .tune_each_group=TRUE.

.force_cv logical. Should models be evaluated across all cross validation slices, even if no hyperparameters are tuned. Default is .force_cv=TRUE.

Details

regress fits all models passed in ... using the m function. The models can be passed as name-function pairs (e.g. ols = m("lm")) or without including a name.

Hyperparameters are tuned automatically using the `.cv` and `.cv_args` arguments, or can be passed to m() (e.g. lasso = m("lasso", lambda = 0.5)). See the individual model functions (?m()) for an overview of hyperparameters.

Cross validation is performed using the 'rsample' package with possible methods including

- 'initial_split' (simple train-test split)
- 'initial_time_split' (train-test split with retained order)
- 'vfold_cv' (aka kfold cross validation)
- 'loo_cv' (leave-one-out)
- 'rolling_origin' (generalized time series cross validation, e.g. rolling or expanding windows)
- 'sliding_window', 'sliding_index', 'sliding_period' (specialized time series splits)
- 'bootstraps'
- 'group_vfold_cv', 'group_bootstraps'

See package documentation for 'rsample' for all available methods.

The mean squared error loss is used to validate performance in the cross validation.

Note that arguments for weights are automatically passed to the functions by setting the `.weights` argument. Weights are also considered during cross validation by calculating weighted versions of the cross validation loss function.

Value

A tidyfit.models frame containing model details for each group.

The `tidyfit.models` frame consists of 4 different components:

1. A group of identifying columns (e.g. model name, data groups, grid IDs)
2. A `model_object` column, which contains the fitted model.
3. A nested `settings` column containing model arguments and hyperparameters
4. Columns showing errors, warnings and messages (if applicable)

Coefficients, predictions, fitted values or residuals can be accessed using the built-in coef, predict, fitted and resid methods. Note that all coefficients are transformed to ensure comparability across methods.
residuals.tidyfit.models

Obtain residuals from models in a tidyfit.models frame

Description

The function generates residuals for all models in a tidyfit.models frame and outputs a tidy frame.

Usage

## S3 method for class ‘tidyfit.models’
residuals(object, ...)

Arguments

  object model.frame created using regress, classify or m

  ... currently not used

Details

The function uses the ‘model_object’ column in a tidyfit.model frame to return residuals for each model.

Value

A 'tibble'.

data <- tidyfit::Factor_Industry_Returns
fit <- regress(data, Return ~ ., m("lasso", lambda = c(0.001, 0.1)), .mask = c("Date", "Industry"))

# Print the models frame
tidyr::unnest(fit, settings)

# View coefficients
coefficients(fit)
Author(s)

Johann Pfitzinger

See Also

\texttt{coef.tidyfit.models}, \texttt{predict.tidyfit.models} and \texttt{fitted.tidyfit.models}

Examples

data <- dplyr::group_by(tidyfit::Factor_Industry_Returns, Industry)
fit <- regress(data, Return ~ ., m("lm"), .mask = "Date")
resid(fit)
Index

* data
  Factor_Industry_Returns, 41
  .model.adalasso, 2, 12, 19, 32, 43
  .model.bayes, 4, 6, 28, 38, 44
  .model.bma, 5, 44
  .model.boost, 7, 30, 36, 43
  .model.chisq, 8, 10, 44
  .model.cor, 9, 9, 44
  .model.enet, 4, 11, 19, 32, 43
  .model.gets, 12, 44
  .model glm, 5, 13, 14, 16, 20, 43
  .model.glmm, 15, 44
  .model.hfr, 16, 44
  .model.lasso, 4, 12, 18, 32, 36, 43
  .model.lm, 14, 19, 28, 33, 35, 43
  .model.mrmr, 21, 29, 44
  .model.mslm, 22, 38, 44
  .model.pcr, 24, 26, 44
  .model.plsr, 17, 25, 25, 44
  .model.quantile, 27, 43
  .model.relief, 28, 44
  .model.rf, 30, 44
  .model.ridge, 4, 12, 19, 31, 43
  .model.robust, 13, 20, 32, 43
  .model subset, 34, 44
  .model.svm, 30, 35, 44
  .model.tvp, 23, 37, 44

classify, 2–4, 7, 8, 11, 14, 15, 18, 21, 28–31, 34, 35, 38, 40, 42, 43, 45, 48
coef.tidyfit.models, 39, 40, 42, 46, 48, 49

data, 41

data, regress, 2, 4, 5, 7, 9, 11, 12, 14–16, 18, 19, 21, 22, 24–32, 34, 35, 37, 39, 40, 42, 43, 45, 46, 48
residuals.tidyfit.models, 41, 42, 46, 48

Factor_Industry_Returns, 41
fitted.tidyfit.models, 41, 42, 46, 49

m, 3–40, 42, 43, 45, 47, 48

predict.tidyfit.models, 39, 41, 42, 45, 48, 49