Package ‘tidyfit’

March 23, 2024

Type Package
Title Regularized Linear Modeling with Tidy Data
Date 2024-03-23
Version 0.7.1
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 7.2.3
URL https://tidyfit.residualmetrics.com,
https://github.com/jpfitzinger/tidyfit
Depends R (>= 3.5)
Imports broom, crayon, dials, dplyr, furrr, generics, magrittr, MASS, methods, progressr, purrr, rlang, rsample, stats, tibble, tidyr, utils, vctrs, yardstick
Suggests arm, bestglm, BMS, BoomSpikeSlab, CORElearn, e1071, gaselect, gets, ggplot2, glmnet, hfr, iml, kableExtra, knitr, lme4, lmtest, lubridate, markdown, mboost, monomvn, mRMRe, MSwM, nnet, pls, quantreg, quantregForest, randomForest, sandwich, sensitivity, shrinkTVP, stringr, rmarkdown, testthat (>= 3.0.0)
VignetteBuilder knitr
Config/testthat/edition 3
NeedsCompilation no
Repository CRAN
Date/Publication 2024-03-23 15:20:02 UTC
## R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>.fit.adalasso</td>
<td>3</td>
</tr>
<tr>
<td>.fit.anova</td>
<td>4</td>
</tr>
<tr>
<td>.fit.bayes</td>
<td>5</td>
</tr>
<tr>
<td>.fit.blm</td>
<td>7</td>
</tr>
<tr>
<td>.fit.bma</td>
<td>8</td>
</tr>
<tr>
<td>.fit.boost</td>
<td>9</td>
</tr>
<tr>
<td>.fit.bridge</td>
<td>11</td>
</tr>
<tr>
<td>.fit.chisq</td>
<td>12</td>
</tr>
<tr>
<td>.fit.cor</td>
<td>13</td>
</tr>
<tr>
<td>.fit.enet</td>
<td>14</td>
</tr>
<tr>
<td>.fit.genetic</td>
<td>16</td>
</tr>
<tr>
<td>.fit.glm</td>
<td>19</td>
</tr>
<tr>
<td>.fit.glmm</td>
<td>20</td>
</tr>
<tr>
<td>.fit.hfr</td>
<td>21</td>
</tr>
<tr>
<td>.fit.lasso</td>
<td>23</td>
</tr>
<tr>
<td>.fit.lm</td>
<td>24</td>
</tr>
<tr>
<td>.fit.mnrfr</td>
<td>26</td>
</tr>
<tr>
<td>.fit.mslm</td>
<td>27</td>
</tr>
<tr>
<td>.fit.nnnet</td>
<td>29</td>
</tr>
<tr>
<td>.fit.pcr</td>
<td>30</td>
</tr>
<tr>
<td>.fit.plsr</td>
<td>32</td>
</tr>
<tr>
<td>.fit.quantile</td>
<td>33</td>
</tr>
<tr>
<td>.fit.quantile_rf</td>
<td>35</td>
</tr>
<tr>
<td>.fit.relief</td>
<td>36</td>
</tr>
<tr>
<td>.fit.rf</td>
<td>38</td>
</tr>
<tr>
<td>.fit.ridge</td>
<td>39</td>
</tr>
<tr>
<td>.fit.robust</td>
<td>40</td>
</tr>
<tr>
<td>.fit.spikeslab</td>
<td>42</td>
</tr>
<tr>
<td>.fitsubset</td>
<td>43</td>
</tr>
<tr>
<td>.fit.svm</td>
<td>45</td>
</tr>
<tr>
<td>.fit.tvp</td>
<td>46</td>
</tr>
<tr>
<td>classify</td>
<td>48</td>
</tr>
<tr>
<td>coef.tidyfit.models</td>
<td>50</td>
</tr>
<tr>
<td>explain</td>
<td>51</td>
</tr>
<tr>
<td>explain.tidyfit.models</td>
<td>53</td>
</tr>
<tr>
<td>Factor_Industry_Returns</td>
<td>55</td>
</tr>
<tr>
<td>fitted.tidyfit.models</td>
<td>55</td>
</tr>
<tr>
<td>m</td>
<td>56</td>
</tr>
<tr>
<td>predict.tidyfit.models</td>
<td>59</td>
</tr>
<tr>
<td>regress</td>
<td>60</td>
</tr>
<tr>
<td>residuals.tidyfit.models</td>
<td>62</td>
</tr>
</tbody>
</table>

Index 64
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'
.fit(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*)
- `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

.fit.lasso, .fit.enet, .fit.ridge and m methods

Examples

# 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)

Description

Performs Analysis of Variance on a 'tidyFit' R6 class. The function can be used with regress or classify.

Usage

## S3 method for class 'anova'
.fit(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::anova`. See `?anova` for more details. First a `glm` model is fitted which is passed to `anova`.

**Value**

A fitted 'tidyFit' class model.

**Author(s)**

Johann Pfitzinger

**See Also**

`.fit.lm`, `.fit.glm` and `m` methods

**Examples**

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

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

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

---

### .fit.bayes

*Bayesian generalized linear regression for tidyfit*

**Description**

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

**Usage**

```r
## S3 method for class 'bayes'
.fit(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`):
The function provides a wrapper for `arm::bayesglm`. See `?bayesglm` for more details.

Implementation
No implementation notes

Value
A fitted `tidyFit` class model.
A `tibble`.

Author(s)
Johann Pfitzinger

References

See Also
.fit(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)
```
Bayesian Lasso regression for tidyfit

Description

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

Usage

```r
## S3 method for class 'blasso'
.fit(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 `monomvn::blasso`. See `?blasso` for more details.

**Implementation**

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

Value

A fitted tidyFit class model.

Author(s)

Johann Pfitzinger

References


See Also

`.fit.lasso`, `.fit.bridge` and `m` methods
Examples

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

# Stand-alone function
fit <- m("blasso", Return ~ ., data, T = 100)
fit

# Within 'regress' function
fit <- regress(data, Return ~ ., m("blasso", T = 100),
               .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'
.fit(bma)
```

**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`)**

- `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
`.fit.bayes` and `m` methods

Examples
```r
# 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'
.fit(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:

- 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)
Bayesian ridge regression for tidyfit

Description

Fits a Bayesian ridge regression on a `tidyFit` R6 class. The function can be used with `regress`.

Usage

```r
## S3 method for class 'bridge'
.fit( 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 dplyr or dplyr).

Details

Hyperparameters:

None. Cross validation not applicable.

Important method arguments (passed to `m`)

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

Implementation

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

Value

A fitted tidyFit class model.

Author(s)

Johann Pfitzinger

References


See Also

`.fit.ridge`, `.fit.bllasso` and `m` methods
Examples

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

# Stand-alone function
fit <- m("bridge", Return ~ ., data, T = 100)
fit

# Within 'regress' function
fit <- regress(data, Return ~ ., m("bridge", T = 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

```r
## S3 method for class 'chisq'
.fit(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::chisq.test`. See `?chisq.test` for more details.

Implementation

Results can be viewed using `coef`.

Value

A fitted ‘tidyFit’ class model.
.fit.cor

Author(s)
Johann Pfitzinger

See Also
.fit.cor and m methods

Examples

# 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)

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

Usage

## S3 method for class 'cor'
.fit(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)
The function provides a wrapper for stats::cor.test. See ?cor.test for more details.

Implementation
Results can be viewed using coef.
ElasticNet regression or classification for tidyFit

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'
.fit(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 (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

.fit.lasso, .fit.adalasso, .fit.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 linear regression with variable selection using a genetic algorithm on a 'tidyFit' R6 class. The function can be used with `regress`.

Usage

```r
# S3 method for class 'genetic'
.fit(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`)

- `statistic`
- `populationSize`
- `numGenerations`
- `minVariables`
- `maxVariables`

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

Implementation

Control arguments are passed to `gaselect::genAlgControl` (the function automatically identifies which arguments are for the control object, and which for `gaselect::genAlg`).

gaselect::evaluatorLM is used as the evaluator with the relevant arguments automatically identified by the function.

Value

A fitted tidyFit class model.

Author(s)

Johann Pfitzinger
References


See Also

.fit.lm, .fit.bayes and m methods

Examples

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

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

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

Description

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

Usage

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

Arguments

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

Hyperparameters:

None. Cross validation not applicable.

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

- \texttt{max\_paths} (Number of paths to search)

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

Implementation

Print output is suppressed by default. Use \texttt{\'print.searchinfo = TRUE\'} for print output.

Value

A fitted \texttt{\'tidyFit\'} class model.

Author(s)

Johann Pfitzinger

References


See Also

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

Examples

# 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)
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'
.fit.glm.
.fit(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

`.fit.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)
```

Generalized linear mixed-effects model for tidyfit

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'
.fit(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.
Author(s)

Johann Pfitzinger

References


See Also

.fit.glm and m methods

Examples

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

# Estimate model with random effects
fit <- classify(data, Return ~ CMA + (CMA | Industry), logit = m("glmm"),
               .mask = "Date")
fit

---

.fit.hfr

Hierarchical feature regression for tidyfit

Description

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

Usage

## S3 method for class 'hfr'
.fit(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 (proportional size of regression graph)

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

The hierarchical feature regression is estimated using the \texttt{hfr::cv.hfr} function. See \texttt{?cv.hfr} for more details.

Implementation

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

If no hyperparameter grid is provided (\texttt{is.null(control$kappa)}), the default is \texttt{seq(0, 1, by = 0.1)}.

Value

A fitted `tidyFit` class model.

Author(s)

Johann Pfitzinger

References

Pfitzinger J (2022). \textit{hfr: Estimate Hierarchical Feature Regression Models}. R package version 0.5.0, \url{https://CRAN.R-project.org/package=hfr}.

See Also

\texttt{.fit.plsr} and \texttt{m} methods

Examples

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

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

# Within 'regress' function
fit <- regress(data, Return ~ ., m("hfr", kappa = 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'.fit
.lasso.fit(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

.fit.enet, .fit.ridge, .fit.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)

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'
.fit(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::lm`. See `?lm` 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

See Also

`.fit.robust`, `fit.glm` and `m` methods

Examples

```r
# 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
```
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'
.fit(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
m methods

Examples
# Load data
data <- tidyfit::Factor_Industry_Returns
data <- dplyr::filter(data, Industry == "HiTec")
data <- dplyr::select(data, SMB, HML, RMW, CMA, Return)

## Not run:
fit <- m("mrmr", Return ~ ., data, feature_count = 2)

# Retrieve selected features
coeff(fit)

## End(Not run)
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**

.fit.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)
```
Description

Fits a single-hidden-layer neural network regression on a 'tidyFit' R6 class. The function can be used with `regress` and `classify`.

Usage

```r
## S3 method for class 'nnet'
.fit.nnet(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:**

- `size` *(number of units in the hidden layer)*
- `decay` *(parameter for weight decay)*
- `maxit` *(maximum number of iterations)*

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

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

Implementation

For `regress`, linear output units (`linout=TRUE`) are used, while `classify` implements the default logic of `nnet` (`entropy=TRUE` for 2 target classes and `softmax=TRUE` for more classes).

Value

A fitted 'tidyFit' class model.

Author(s)

Phil Holzmeister
Examples

# Load data
data <- tidyfit::Factor_Industry_Returns

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

# Within 'regress' function
fit <- regress(data, Return ~ ., m("nnet", decay=0.5, size = 8),
               .mask = c("Date", "Industry"))

# Within 'classify' function
fit <- classify(iris, Species ~ ., m("nnet", decay=0.5, size = 8))

---

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'
.fit(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` (<i>number of components</i>)
- `ncomp_pct` (<i>number of components, percentage of features</i>)

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

Liland K, Mevik B, Wehrens R (2022). pls: Partial Least Squares and Principal Component Re-

See Also

.fit.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)
Partial Least Squares Regression for tidyfit

Description

Fits a partial least squares regression on a 'tidyFit' R6 class. The function can be used with `regress`.

Usage

```r
## S3 method for class 'plsr'
.fit.plsr
 .fit(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:**

- `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
.fit.quantile

References

See Also
.fit.pcr 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("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)

---

.fit.quantile 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'
.fit(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)

• 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

.fit.lm, .fit.bayes and m methods

Examples

# Load data
data <- tidyfit::Factor_Industry_Returns

fit <- regress(data, Return ~ ., 
    m("quantile", tau = c(0.1, 0.5, 0.9)),
    .mask = c("Date", "Industry"))

coef(fit)
Quantile regression forest for tidyfit

Description

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

Usage

```r
## S3 method for class 'quantile_rf'
.fit.quantile_rf

fit(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:

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

Important method arguments (passed to `m`)

- `tau` (the quantile(s) to be estimated)

The function provides a wrapper for `quantregForest::quantregForest`. See `?quantregForest` for more details. The argument `tau` is the chosen quantile (default `tau = 0.5`). `tau` is passed directly to `m('quantile_rf', tau = c(0.1, 0.5, 0.9))` and is not passed to `predict` as in the `quantregForest::quantregForest` package. This is done to ensure a consistent interface with the quantile regression from `quantreg`.

Implementation

No implementation notes

Value

A fitted 'tidyFit' class model.

Author(s)

Johann Pfitzinger
References


See Also

.fit.quantile, .fit.rf 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("quantile_rf", Return ~ ., data, tau = 0.5, ntree = 50)
fitted(fit)

# Within 'regress' function
fit <- regress(data, Return ~ .,
m("quantile_rf", tau = c(0.1, 0.5, 0.9), ntree = 50))
explain(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'
.fit(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 \texttt{m})

- estimator (selection algorithm to use (default is 'ReliefFequalK'))

The ReliefF algorithm is estimated using the \texttt{CORElearn::attrEval} function. See \texttt{?attrEval} for more details.

Implementation

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

The Relief objects have no \texttt{predict} and related methods.

Value

A fitted 'tidyFit' class model.

Author(s)

Johann Pfitzinger

References


See Also

\texttt{.fit.mrmr} and \texttt{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("relief", Return ~ ., data)
coef(fit)

# Within 'regress' function
fit <- regress(data, Return ~ ., m("relief"))
coef(fit)
```
Random Forest regression or classification for tidyfit

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'
.fit(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:
- `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
.fit.svm, .fit.boost 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("rf", Return ~ ., data)
fit

# Within 'regress' function
fit <- regress(data, Return ~ ., m("rf"))
explain(fit)

---

.fit.ridge

Ridge regression and classification for tidyfit

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

## S3 method for class 'ridge'.fit(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

.fit.lasso, .fit.adalasso, .fit.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)

Description

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

Usage

## S3 method for class 'robust'
.fit(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)**

  • `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

`.fit.lm` and `m` methods

Examples

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

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")
tidyrr::unnest(coef(fit), model_info)
```
Bayesian Spike and Slab regression or classification for tidyfit

Description

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

Usage

```r
## S3 method for class 'spikeslab'
.fit(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)**

In the case of `regression`, arguments are passed to `BoomSpikeSlab::lm.spike` and `BoomSpikeSlab::SpikeSlabPrior`. Check those functions for details.

- `BoomSpikeSlab::SpikeSlabPrior`
  - `expected.r2`
  - `prior.df`
  - `expected.model.size`

- `BoomSpikeSlab::lm.spike`
  - `niter`

In the case of `classification`, arguments are passed to `BoomSpikeSlab::logit.spike` and `BoomSpikeSlab::SpikeSlabGlmPrior`. Check those functions for details.

- `BoomSpikeSlab::logit.spike`
  - `niter`

I advise against the use of `BoomSpikeSlab::SpikeSlabGlmPrior` at the moment, since it appears to be buggy.

The function provides wrappers for `BoomSpikeSlab::lm.spike` and `BoomSpikeSlab::logit.spike`. See `?lm.spike` and `?logit.spike` for more details.
Implementation
Prior arguments are passed to BoomSpikeSlab::SpikeSlabPrior and BoomSpikeSlab::SpikeSlabGlmPrior (the function automatically identifies which arguments are for the prior, and which for BoomSpikeSlab::lm.spike or BoomSpikeSlab::logit.spike).
BoomSpikeSlab::logit.spike is automatically selected when using classify.

Value
A fitted tidyFit class model.

Author(s)
Johann Pfitzinger

References

See Also
.fit.lasso, .fit.blasso and m methods

Examples
# Load data
data <- tidyfit::Factor_Industry_Returns

# Stand-alone function
fit <- m("spikeslab", Return ~ ., data, niter = 100)
fit

# Within 'regress' function
fit <- regress(data, Return ~ ., m("spikeslab", niter = 100),
               .mask = c("Date", "Industry"))
coef(fit)

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'
.fit(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`)

- `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

`.fit.lm` and `m` methods
Examples

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

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

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

Description

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

Usage

```
## S3 method for class 'svm'
.fit(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:

- 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.
.fit.tvp

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'
.fit(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


See Also

`.fit.bayes`, `.fit.mslm` and `m` methods

Examples

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

# 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)
```
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`.
- `.return_grid` logical. Should the output of the individual hyperparameter grids be returned or only the best fitting set of hyperparameters. Default is `.return_grid=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=FALSE`.

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)
classify

- ‘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 underlying 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)
 coef.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

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

explain

An interface for variable importance measures for a fitted tidyfit.models frames

Description

A generic method for calculating XAI and variable importance methods for tidyfit.models frames.

Usage

`explain(object, use_package = NULL, use_method = NULL, ...)`

Arguments

- `object` model.frame created using `regress`, `classify` or `m`
- `use_package` the package to use to calculate variable importance. See 'Details' for possible options.
- `use_method` the method from 'use_package' that should be used to calculate variable importance.
- `...` additional arguments passed to the importance method

Details

**WARNING** This function is currently in an experimental stage.

The function uses the 'model_object' column in a tidyfit.model frame to return variable importance measures for each model.

Possible packages and methods include:
sensitivity package::
The package provides methods to assess variable importance in linear regressions (`lm`) and classifications (`glm`).

Usage: `use_package = "sensitivity"`

Methods:
- "lmg" (Shapley regression),
- "pmvd" (Proportional marginal variance decomposition),
- "src" (standardized regression coefficients),
- "pcc" (partial correlation coefficients),
- "johnson" (Johnson indices)

See `?sensitivity::lmg` for more information and additional arguments.

iml package::
Integration with iml is currently in progress. The methods can be used for `nnet`, `rf`, `lasso`, `enet`, `ridge`, `adalasso`, `glm` and `lm`.

Usage: `use_package = "iml"`

Methods:
- "Shapley" (SHAP values)
- "LocalModel" (LIME)
- "FeatureImp" (Permutation-based feature importance)

The argument `which_rows` (vector of integer indexes) can be used to explain specific rows in the data set for Shapley and LocalModel methods.

randomForest package::
This uses the native importance method of the randomForest package and can be used with `rf` and `quantile_rf` regression and classification.

Usage: `use_package = "randomForest"`

Methods:
- "mean_decrease_accuracy"

Value
A `tibble`.

Author(s)
Johann Pfitzinger

References


Explain.TidyFit.Models

Examples

data <- dplyr::group_by(tidyfit::Factor_Industry_Returns, Industry)
fit <- regress(data, Return ~ ., m("lm"), .mask = "Date")
explain(fit, use_package = "sensitivity", use_method = "src")

data <- dplyr::filter(tidyfit::Factor_Industry_Returns, Industry == Industry[1])
fit <- regress(data, Return ~ ., m("lm"), .mask = c("Date", "Industry"))
explain(fit, use_package = "iml", use_method = "Shapley", which_rows = c(1))

Explain.TidyFit.Models

An interface for variable importance measures for a fitted tidyfit.models frames

Description

A generic method for calculating XAI and variable importance methods for tidyfit.models frames.

Usage

## S3 method for class 'tidyfit.models'
explain(
  object,
  use_package = NULL,
  use_method = NULL,
  ..., .keep_grid_id = FALSE
)

Arguments

object model.frame created using `regress, classify` or `m`

use_package the package to use to calculate variable importance. See 'Details' for possible options.

use_method the method from 'use_package' that should be used to calculate variable importance.

... additional arguments passed to the importance method

.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

**WARNING** This function is currently in an experimental stage.

The function uses the 'model_object' column in a tidyfit.model frame to return variable importance measures for each model.

**Possible packages and methods include:**

**sensitivity package::**

The package provides methods to assess variable importance in linear regressions ('lm') and classifications ('glm').

*Usage:* `use_package="sensitivity" Methods:*

- "lmg" (Shapley regression),
- "pmvd" (Proportional marginal variance decomposition),
- "src" (standardized regression coefficients),
- "pcc" (partial correlation coefficients),
- "johnson" (Johnson indices)

See ?sensitivity::lmg for more information and additional arguments.

**iml package::**

Integration with iml is currently in progress. The methods can be used for 'nnnet', 'rf', 'lasso', 'enet', 'ridge', 'adalasso', 'glm' and 'lm'.

*Usage:* `use_package="iml" Methods:*

- "Shapley" (SHAP values)
- "LocalModel" (LIME)
- "FeatureImp" (Permutation-based feature importance)

The argument 'which_rows' (vector of integer indexes) can be used to explain specific rows in the data set for Shapley and LocalModel methods.

**randomForest package::**

This uses the native importance method of the randomForest package and can be used with 'rf' and 'quantile_rf' regression and classification.

*Usage:* `use_package="randomForest" Methods:*

- "mean_decrease_accuracy"

**Value**

A 'tibble'.

**Author(s)**

Johann Pfitzinger
Factor_Industry_Returns

References

Examples
```r
data <- dplyr::group_by(tidyfit::Factor_Industry_Returns, Industry)
fit <- regress(data, Return ~ ., m("lm"), .mask = "Date")
explain(fit, use_package = "sensitivity", use_method = "src")
data <- dplyr::filter(tidyfit::Factor_Industry_Returns, Industry == Industry[1])
fit <- regress(data, Return ~ ., m("lm"), .mask = c("Date", "Industry"))
explain(fit, use_package = "iml", use_method = "Shapley", which_rows = c(1))
```

Factor_Industry_Returns

*Industry-Factor Returns Data Set*

Description
The data set includes monthly returns between 1963 and 2022 for 10 industries, as well as factor values for 5 Fama-French factors.

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

```r
## 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

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

---

`m`  
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

```r
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 `.fit.lm` for details.
- "glm" performs a generalized regression or classification using `stats::glm`. See `.fit.glm` for details.
- "anova" performs analysis of variance using `stats::anova`. See `.fit.anova` for details.
- "robust" performs a robust regression using `MASS::rlm`. See `.fit.robust` for details.
- "quantile" performs a quantile regression using `quantreg::rq`. See `.fit.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 `.fit.lasso` for details.
- "ridge" performs a linear regression or classification with L2 penalty using `glmnet::glmnet`. See `.fit.ridge` for details.
- "adalasso" performs an Adaptive Lasso regression or classification using `glmnet::glmnet`. See `.fit.adalasso` for details.
- "enet" performs a linear regression or classification with L1 and L2 penalties using `glmnet::glmnet`. See `.fit.enet` for details.

**Other Machine Learning:**
- "boost" performs gradient boosting regression or classification using `mboost::glmboost`. See `.fit.boost` for details.
- "rf" performs a random forest regression or classification using `randomForest::randomForest`. See `.fit.rf` for details.
- "svm" performs a support vector regression or classification using `e1071::svm`. See `.fit.svm` for details.
- "nnet" performs a neural network regression or classification using `nnet::nnet`. See `.fit.nnet` for details.

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

"subset" performs a best subset regression or classification using `bestglm::bestglm` (wrapper for leaps). See `.fit.subset` for details.

"gets" performs a general-to-specific regression using `gets::gets`. See `.fit.gets` for details.

**Bayesian methods:**

"bayes" performs a Bayesian generalized regression or classification using `arm::bayesglm`. See `.fit.bayes` for details.

"bridge" performs a Bayesian ridge regression using `monomvn::bridge`. See `.fit.bridge` for details.

"blasso" performs a Bayesian Lasso regression using `monomvn::blasso`. See `.fit.blasso` for details.

"spikeslab" performs a Bayesian Spike and Slab regression using `BoomSpikeSlab::lm.spike`. See `.fit.spikeslab` for details.

"bma" performs a Bayesian model averaging regression using `BMS::bms`. See `.fit.bma` for details.

"tvp" performs a Bayesian time-varying parameter regression using `shrinkTVP::shrinkTVP`. See `.fit.tvp` for details.

**Mixed-effects modeling:**

"glmm" performs a mixed-effects GLM using `lme4::glmer`. See `.fit.glmm` for details.

**Specialized time series methods:**

"mslm" performs a Markov-switching regression using `MSwM::msmFit`. See `.fit.mslm` for details.

**Feature selection:**

"cor" calculates Pearson’s correlation coefficient using `stats::cor.test`. See `.fit.cor` for details.

"chisq" calculates Pearson’s Chi-squared test using `stats::chisq.test`. See `.fit.chisq` for details.

"mrmr" performs a minimum redundancy, maximum relevance features selection routine using `mRMR::mRMR.ensemble`. See `.fit.mrmr` for details.

"relief" performs a ReliefF feature selection routine using `CORElearn::attrEval`. See `.fit.relief` for details.

"genetic" performs a linear regression with feature selection using the genetic algorithm implemented in `gaselect::genAlg`. See `.fit.genetic` 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
predict.tidyfit.models

See Also

regress and classify methods

Examples

# Load data
data <- tidyfit::Factor_Industry_Returns

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

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

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 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`.
- `.return_grid` logical. Should the output of the individual hyperparameter grids be returned or only the best fitting set of hyperparameters. Default is `.return_grid=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=FALSE`.
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.

Author(s)

Johann Pfitzinger

See Also

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

Obtain residuals from models in a tidyfit.models frame

Examples

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
tidyrr::unnest(fit, settings)

# View coefficients
coeff(fit)

residuals.tidyfit.models

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'.

Author(s)

Johann Pfitzinger

See Also

coeff.tidyfit.models, predict.tidyfit.models and 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, 55
  .fit.adalasso, 3, 15, 24, 40, 57
  .fit.anova, 4, 57
  .fit.bayes, 5, 9, 17, 34, 47, 58
  .fit.blasso, 7, 11, 43, 58
  .fit.bma, 8, 58
  .fit.boost, 9, 38, 46, 57
  .fit.bridge, 7, 11, 58
  .fit.chisq, 12, 14, 58
  .fit.cor, 13, 13, 58
  .fit.enet, 4, 14, 24, 40, 57
  .fit.fivefold, 16, 58
  .fit.gets, 17, 58
  .fit.glm, 5, 6, 18, 19, 21, 25, 57
  .fit.glmm, 20, 58
  .fit.hfr, 21, 57
  .fit.lasso, 4, 7, 15, 23, 40, 43, 46, 57
  .fit.lm, 5, 17, 19, 24, 34, 41, 44, 57
  .fit.mrmlr, 26, 37, 58
  .fit.mslm, 27, 47, 58
  .fit.nnet, 29, 57
  .fit.pcr, 30, 33, 57
  .fit.plsr, 22, 31, 32, 57
  .fit.quantile, 33, 36, 57
  .fit.quantile_rf, 35
  .fit.relief, 36, 58
  .fit.rf, 36, 38, 57
  .fit.ridge, 4, 11, 15, 24, 39, 57
  .fit.robust, 18, 25, 40, 57
  .fit.spikeslab, 42, 58
  .fit.subset, 43, 58
  .fit.svm, 38, 45, 57
  .fit.tvp, 28, 46, 58

classify, 3–5, 9, 12, 14, 15, 19, 20, 23, 26, 29, 36–39, 42, 43, 45, 48, 50, 51, 53, 56, 59, 61, 62
coef.tidyfit.models, 49, 50, 56, 60–62

data, 5
explain, 51
explain.tidyfit.models, 53
fitted.tidyfit.models, 51, 55, 60, 62
m, 3–48, 50, 51, 53, 56, 59, 61, 62
predict.tidyfit.models, 49, 51, 56, 59, 61, 62
regress, 3–5, 7–9, 11, 13, 14, 16, 17, 19–21, 23, 24, 26, 27, 29–33, 35–40, 42, 43, 45, 46, 49–51, 53, 56, 59, 60, 62
residuals.tidyfit.models, 51, 56, 60, 62

64