Package 'funcml'

April 21, 2026

Type Package

Title Functional Machine Learning Framework

Version 0.7.1

Depends R (>= 3.5.0)

Description A compact and explicit machine learning framework for supervised learning, resampling-based evaluation, hyperparameter tuning, learner comparison, interpretation, and plug-in g-computation. The package uses standard formulas for model specification and provides stable S3 interfaces for fitting, evaluation, tuning, interpretation, and causal estimation across a learner registry with multiple backend engines. Implemented interpretation methods build on established approaches such as permutation-based variable importance, partial dependence, individual conditional expectation, accumulated local effects, SHAP, and LIME; see Friedman (2001) <doi:10.1214/aos/1013203451>, Goldstein et al. (2015) <doi:10.1080/10618600.2014.907095>, Apley and Zhu (2020) <doi:10.1111/rssb.12377>, Lundberg and Lee (2017) <doi:10.48550/arXiv.1705.07874>, and Ribeiro et al. (2016) <doi:10.48550/arXiv.1602.04938>. The framework is intentionally opinionated: preprocessing is expected to occur outside the modeling step, and the API emphasizes explicit inputs, consistent object contracts, and compact interfaces rather than feature-by-feature competition with larger machine learning ecosystems.

License GPL-3

URL https://github.com/ielbadisy/funcml

BugReports https://github.com/ielbadisy/funcml/issues

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

Imports stats, utils, methods, ggplot2, functionals, grDevices, tools, MASS, mgcv, nnet, rpart, glmnet, ranger, e1071, randomForest, gbm, C50, kknn, earth, naivebayes, mda, ada, pls, partykit, dbarts, xgboost, lightgbm, shapviz

Suggests testthat (>= 3.1.0), knitr, rmarkdown, roxygen2, gggenes, ggfittext

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Imad El Badisy [aut, cre]

Maintainer Imad El Badisy <elbadisyimad@gmail.com>

Repository CRAN

Date/Publication 2026-04-21 21:12:15 UTC

Contents

`arthritis` *Arthritis survey data*

Description

A classification dataset on arthritis status and related demographic and behavioral covariates.

Usage

```
arthritis
```

Format

A data frame with 4,856 rows and 12 variables:

id Participant identifier.

status Arthritis status (`"Yes"` or `"No"`).

heart_attack_relative Whether a relative had a heart attack.

gender Participant gender.

age Participant age in years.

bmi Body mass index.

diabetes Whether the participant has diabetes.

alcohol Whether the participant reports alcohol use.

smoke Whether the participant smokes.

prehypertension Whether the participant has prehypertension.

vegetarian Whether the participant follows a vegetarian diet.

covered_health Whether the participant has health coverage.

Details

Column names were standardized to `snake_case` when packaging the data.

Source

Original arthritis survey dataset distributed with the project materials.

Examples

```
str(funcml::arthritis)
table(funcml::arthritis$status)
```

`bangladeshmaternalrisk`

Bangladesh maternal risk data

Description

A classification dataset for maternal health risk level with vital signs, diabetes history, and related clinical indicators.

Usage

```
bangladeshmaternalrisk
```

Format

A data frame with 1,205 rows and 12 variables:

age Maternal age in years.

systolic_bp Systolic blood pressure.

diastolic Diastolic blood pressure.

bs Blood sugar measurement.

body_temp Body temperature.

bmi Body mass index.

previous_complications Indicator for previous pregnancy complications.

preexisting_diabetes Indicator for preexisting diabetes.

gestational_diabetes Indicator for gestational diabetes.

mental_health Indicator for mental health concerns.

heart_rate Heart rate.

risk_level Maternal risk level outcome.

Details

Column names were standardized to `snake_case` when packaging the data.

Source

Mojumdar MU, Sarker D, Assaduzzaman M, et al. (2025). Maternal health risk factors dataset: Clinical parameters and insights from rural Bangladesh. *Data in Brief*, 59(Suppl 2), 111363. doi:10.1016/j.dib.2025.111363.

Examples

```
str(funcml::bangladeshmaternalrisk)
table(funcml::bangladeshmaternalrisk$risk_level)
```

---

birthweight *Birth weight data*

Description

A regression-oriented birth weight dataset with maternal risk factors and a derived low-birth-weight indicator.

Usage

```
birthweight
```

Format

A data frame with 189 rows and 10 variables:

age Maternal age in years.

lwt Maternal weight at the last menstrual period.

race Maternal race code.

smoke Smoking status indicator.

ptl Number of previous premature labors.

ht History of hypertension indicator.

ui Presence of uterine irritability indicator.

ftv Number of physician visits in the first trimester.

birth_weight_g Birth weight in grams.

low_birth_weight Low-birth-weight outcome indicator.

Source

Hosmer DW, Lemeshow S (1989). *Applied Logistic Regression*. Wiley. The packaged data are a lightly renamed version of the classic `MASS::birthwt` dataset.

Examples

```
str(funcml::birthweight)
summary(funcml::birthweight$birth_weight_g)
```

`breastcancerdiagnostic` *Breast cancer diagnostic data*

Description

A binary classification dataset for breast cancer diagnosis using tumor morphology measurements.

Usage

```
breastcancerdiagnostic
```

Format

A data frame with 569 rows and 31 variables:

radius_mean Mean radius.

texture_mean Mean texture.

perimeter_mean Mean perimeter.

area_mean Mean area.

smoothness_mean Mean smoothness.

compactness_mean Mean compactness.

concavity_mean Mean concavity.

concave_pts_mean Mean number of concave points.

symmetry_mean Mean symmetry.

fractal_dim_mean Mean fractal dimension.

radius_se Radius standard error.

texture_se Texture standard error.

perimeter_se Perimeter standard error.

area_se Area standard error.

smoothness_se Smoothness standard error.

compactness_se Compactness standard error.

concavity_se Concavity standard error.

concave_pts_se Concave points standard error.

symmetry_se Symmetry standard error.

fractal_dim_se Fractal dimension standard error.

radius_worst Worst radius.

texture_worst Worst texture.

perimeter_worst Worst perimeter.

area_worst Worst area.

smoothness_worst Worst smoothness.

compactness_worst Worst compactness.

concavity_worst Worst concavity.

concave_pts_worst Worst number of concave points.

symmetry_worst Worst symmetry.

fractal_dim_worst Worst fractal dimension.

diagnosis Diagnosis outcome (`"B"` = benign, `"M"` = malignant).

Details

Column names were standardized to `snake_case` when packaging the data.

Source

Breast Cancer Wisconsin Diagnostic Dataset from the UCI Machine Learning Repository, packaged in `dslabs::brca`.

Examples

```
str(funcml::breastcancerdiagnostic)
table(funcml::breastcancerdiagnostic$diagnosis)
```

`breastcancerwisconsin` *Wisconsin breast cancer data*

Description

A binary classification dataset for breast cancer diagnosis from cytology measurements.

Usage

```
breastcancerwisconsin
```

Format

A data frame with 699 rows and 10 variables:

clump_thickness Clump thickness score.

uniformity_cell_size Uniformity of cell size score.

uniformity_cell_shape Uniformity of cell shape score.

marginal_adhesion Marginal adhesion score.

single_epithelial_cell_size Single epithelial cell size score.

bare_nuclei Bare nuclei score.

bland_chromatin Bland chromatin score.

normal_nucleoli Normal nucleoli score.

mitoses Mitoses score.

class Diagnostic class (`2` = benign, `4` = malignant).

Source

Wisconsin Breast Cancer Database from University of Wisconsin Hospitals, distributed through the UCI Machine Learning Repository.

Examples

```
str(funcml::breastcancerwisconsin)
table(funcml::breastcancerwisconsin$class)
```

---

cancerremission *Cancer remission data*

Description

A binary classification dataset on cancer remission status using leukemia index and treatment group indicators.

Usage

```
cancerremission
```

Format

A data frame with 27 rows and 3 variables:

li Leukemia index measurement.

m Treatment group indicator.

remission Remission outcome indicator (`0` = no remission, `1` = remission).

Details

Column names were standardized to `snake_case` when packaging the data.

Source

Davison AC, Hinkley DV (1997). *Bootstrap Methods and Their Application*. Cambridge University Press. The packaged data are from `boot::remission`.

Examples

```
str(funcml::cancerremission)
table(funcml::cancerremission$remission)
```

cd4counts *CD4 follow-up data*

Description

A regression dataset relating baseline CD4 counts to one-year follow-up CD4 measurements in HIV-positive patients.

Usage

```
cd4counts
```

Format

A data frame with 20 rows and 2 variables:

baseline Baseline CD4 count.

oneyear One-year follow-up CD4 count.

Source

Davison AC, Hinkley DV (1997). *Bootstrap Methods and Their Application*. Cambridge University Press. The packaged data are from `boot::cd4`.

Examples

```
str(funcml::cd4counts)
summary(funcml::cd4counts$oneyear)
```

cigsmoke *Youth tobacco survey data*

Description

A classification-oriented survey dataset on smoking exposure, tobacco use, and tobacco-related environments among youth respondents.

Usage

```
cigsmoke
```

Format

A data frame with 3,915 rows and 27 variables:

final_wgt Survey final weight.

age Age group.

gender Gender.

income Personal spending money category.

parent_work Parental work status.

father_education Father's education level.

mother_education Mother's education level.

living_env Living environment.

age_first_cig Age at first cigarette.

cigar_use Cigar use indicator.

noncig_use Non-cigarette tobacco use indicator.

smokeless_use Smokeless tobacco use indicator.

parent_smoke Parental smoking exposure.

friends_smoke Friends' smoking exposure.

home_shs Secondhand smoke exposure at home.

outside_shs Secondhand smoke exposure outside the home.

indoor_ban Indoor smoking ban indicator.

outdoor_ban Outdoor smoking ban indicator.

antitobacco_media Exposure to antitobacco media.

antitobacco_school Exposure to school antitobacco education.

tobacco_media Exposure to tobacco media.

offer_freetobacco Whether free tobacco was offered.

own_items Ownership of tobacco-branded items.

knowledge_harm Knowledge that tobacco is harmful.

e_cig Electronic cigarette use indicator.

stratum Survey stratum identifier.

psu Primary sampling unit identifier.

Details

Column names were standardized to `snake_case` when packaging the data.

Source

Morocco Global Youth Tobacco Survey public-use survey data.

References

Kim N, Loh WY, McCarthy DE (2021). Machine learning models of tobacco susceptibility and current use among adolescents from 97 countries in the Global Youth Tobacco Survey, 2013-2017. *PLOS Global Public Health*, 1(12), e0000060. doi:10.1371/journal.pgph.0000060.

Examples

```
str(funcml::cigsmoke)
table(funcml::cigsmoke$e_cig)
```

---

compare-methods *Methods for learner comparison results.*

Description

These methods provide the standard `print()`, `summary()`, and `plot()` interfaces for `funcml_compare` objects.

Usage

```
## S3 method for class 'funcml_compare'
print(x, ...)

## S3 method for class 'funcml_compare'
summary(object, ...)

## S3 method for class 'funcml_compare'
plot(x, ...)
```

Arguments

`x`	A `funcml_compare` object.
`...`	Additional arguments passed to the underlying method.
`object`	A `funcml_compare` object.

Value

`print()` and `summary()` return the input object or results table invisibly. `plot()` returns a `ggplot2` object.

Examples

```
cmp <- compare_learners(
  data = mtcars,
  formula = mpg ~ wt + hp,
  models = c("glm", "rpart"),
  resampling = cv(3, seed = 1),
```

```
  metrics = c("rmse", "mae")
)
print(cmp)
summary(cmp)
plot(cmp)
```

`compare_learners` *Compare multiple learners with optional tuning.*

Description

Compare multiple learners with optional tuning.

Usage

```
compare_learners(
  data,
  formula,
  models,
  specs = NULL,
  resampling = cv(5),
  metrics = NULL,
  type = NULL,
  conf_level = 0.95,
  seed = NULL,
  ncores = NULL,
  tune = FALSE,
  grids = NULL,
  metric = NULL,
  ...
)
```

Arguments

`data`	Data frame.
`formula`	Model formula.
`models`	Character vector of learner ids.
`specs`	Optional named list of fixed specs per learner.
`resampling`	Resampling object from `cv()`.
`metrics`	Character vector of metrics to report. When `tune = TRUE`, these are computed for each learner's tuned best configuration.
`type`	Prediction type override.
`conf_level`	Confidence level for learner summary intervals.
`seed`	Optional seed.

ncores	Optional number of CPU cores used to compare learners. NULL or 1 runs sequentially.
tune	Logical; if TRUE, run tune() for each learner before comparing.
grids	Optional tuning grids. Supply either a single data frame to reuse across learners or a named list of data frames keyed by learner id.
metric	Optimization metric used when tune = TRUE.
...	Additional arguments passed to evaluate() or tune() / fit().

Value

A funcml_compare object.

Examples

```
cmp <- compare_learners(
  data = mtcars,
  formula = mpg ~ wt + hp,
  models = c("glm", "rpart"),
  resampling = cv(3, seed = 1),
  metrics = c("rmse", "mae")
)
cmp$results
```

---

cv *Resampling specification generator.*

Description

Resampling specification generator.

Usage

```
cv(
  v = 5,
  repeats = 1,
  strata = TRUE,
  seed = NULL,
  method = c("vfold", "holdout", "group_vfold", "time"),
  prop = 0.8,
  group = NULL,
  time = NULL,
  initial = NULL,
  assess = NULL,
  skip = 0,
  cumulative = TRUE
)
```

Arguments

v	Number of folds for cross-validation.
repeats	Number of repeats for standard or grouped cross-validation.
strata	Logical; stratify classification outcomes when supported.
seed	Optional seed for reproducibility.
method	Resampling strategy: "vfold", "holdout", "group_vfold", or "time".
prop	Training-set proportion for holdout splits.
group	Optional grouping variable name or vector for grouped CV.
time	Optional ordering variable name or vector for time-aware splits.
initial	Initial training window size for time-aware CV.
assess	Assessment window size for time-aware CV.
skip	Number of observations to skip between successive time splits.
cumulative	Logical; use an expanding training window for time-aware CV.

Value

A funcml_cv object containing fold indices and parameters.

Examples

```
cv(v = 3, repeats = 2, seed = 1)
```

---

doctorvisits *Doctor visits data*

Description

A regression dataset on annual doctor visit counts and related health, demographic, and insurance covariates.

Usage

```
doctorvisits
```

Format

A data frame with 5,190 rows and 12 variables:

visits Number of doctor visits.

gender Recorded gender.

age Age in years scaled to decades.

income Income measure scaled by household composition.

illness Number of illnesses in the previous two weeks.

reduced Number of days with reduced activity.

health Self-rated health score.

private Private insurance indicator.

freepoor Free care indicator for low-income patients.

freerepat Free care indicator for pensioners or veterans.

nchronic Indicator for no chronic condition.

lchronic Indicator for limiting chronic condition.

Source

Cameron AC, Trivedi PK (1998). *Regression Analysis of Count Data*. Cambridge University Press. The packaged data are from `AER::DoctorVisits`.

Examples

```
str(funcml::doctorvisits)
summary(funcml::doctorvisits$visits)
```

`estimate` *Causal effect estimation via plug-in g-computation.*

Description

Causal effect estimation via plug-in g-computation.

Usage

```
estimate(
  data,
  formula,
  model = NULL,
  treatment = NULL,
  estimand = c("ATE", "ATT", "CATE", "IATE"),
  newdata = NULL,
  treatment_level = NULL,
  control_level = NULL,
  spec = NULL,
  type = NULL,
  interval = c("normal", "bootstrap"),
  conf_level = 0.95,
  n_boot = 200,
  seed = NULL,
  fit = NULL,
  ...
)
```

Arguments

data	Data frame.
formula	Outcome model formula. The first term on the right-hand side is treated as the treatment variable unless treatment is supplied.
model	Learner id (ignored if fit supplied).
treatment	Optional treatment variable name.
estimand	One of "ATE", "ATT", "CATE", or "IATE".
newdata	Optional target population for estimand = "CATE" or "IATE".
treatment_level	Optional treated level for binary treatment.
control_level	Optional control level for binary treatment.
spec	Hyperparameter list passed to fit().
type	Prediction type override for the outcome model.
interval	Interval method: "normal" or "bootstrap".
conf_level	Confidence level for uncertainty intervals.
n_boot	Number of bootstrap resamples used when interval = "bootstrap".
seed	Optional seed.
fit	Optional preconfigured funcml_fit object.
...	Passed to fit().

Value

A funcml_estimand object.

Examples

```
causal_data <- mtcars
causal_data$am <- factor(causal_data$am, labels = c("auto", "manual"))
ate <- estimate(
  data = causal_data,
  formula = mpg ~ am + wt + hp,
  model = "glm",
  treatment = "am",
  estimand = "ATE"
)
ate$estimate
```

`estimate-methods` *Methods for causal estimand results.*

Description

These methods provide the standard `print()`, `summary()`, and `plot()` interfaces for `funcml_estimand` objects.

Usage

```
## S3 method for class 'funcml_estimand'
print(x, ...)

## S3 method for class 'funcml_estimand'
summary(object, ...)

## S3 method for class 'funcml_estimand'
plot(x, ...)
```

Arguments

`x`	A `funcml_estimand` object.
`...`	Additional arguments passed to the underlying method.
`object`	A `funcml_estimand` object.

Value

`print()` and `summary()` return the input object or summary table invisibly. `plot()` returns a `ggplot2` object.

Examples

```
causal_data <- mtcars
causal_data$am <- factor(causal_data$am, labels = c("auto", "manual"))
ate <- estimate(
  data = causal_data,
  formula = mpg ~ am + wt + hp,
  model = "glm",
  treatment = "am",
  estimand = "ATE"
)
print(ate)
summary(ate)
plot(ate)
```

`evaluate` *Cross-validated evaluation.*

Description

Cross-validated evaluation.

Usage

```
evaluate(
  data,
  formula,
  model = NULL,
  spec = NULL,
  resampling = cv(5),
  metrics = NULL,
  type = NULL,
  conf_level = 0.95,
  seed = NULL,
  fit = NULL,
  ncores = NULL,
  ...
)
```

Arguments

`data`	Data frame.
`formula`	Model formula.
`model`	Learner id (ignored if `fit` supplied).
`spec`	Hyperparameter list.
`resampling`	Resampling object from `cv()`.
`metrics`	Character vector of metric names.
`type`	Prediction type override.
`conf_level`	Confidence level for fold-based summary intervals.
`seed`	Optional seed.
`fit`	Optional preconfigured `funcml_fit` object (re-fit per fold).
`ncores`	Optional number of CPU cores used to evaluate resampling folds. `NULL` or `1` runs sequentially.
`...`	Passed to `fit()`.

Value

A `funcml_eval` object.

Examples

```
eval_obj <- evaluate(
  data = mtcars,
  formula = mpg ~ wt + hp,
  model = "glm",
  resampling = cv(3, seed = 1),
  metrics = c("rmse", "mae")
)
eval_obj$summary
```

`evaluate-methods` *Methods for cross-validation results.*

Description

These methods provide the standard `print()` and `summary()` interfaces for `funcml_eval` objects, plus a `plot()` method for fold-level diagnostics.

Usage

```
## S3 method for class 'funcml_eval'
print(x, ...)

## S3 method for class 'funcml_eval'
summary(object, ...)

## S3 method for class 'funcml_eval'
plot(x, ...)
```

Arguments

`x`	A `funcml_eval` object.
`...`	Additional arguments passed to the underlying method.
`object`	A `funcml_eval` object.

Value

`print()` and `summary()` return the input object or summary table invisibly. `plot()` returns a `ggplot2` object.

Examples

```
eval_obj <- evaluate(
  data = mtcars,
  formula = mpg ~ wt + hp,
  model = "glm",
  resampling = cv(3, seed = 1),
  metrics = c("rmse", "mae")
```

```
)
print(eval_obj)
summary(eval_obj)
plot(eval_obj)
```

`fit` *Fit a model using the funcml interface.*

Description

Registered learner ids currently include: regression and classification: `glm`, `rpart`, `glmnet`, `ranger`, `nnet`, `e1071_svm`, `randomForest`, `gbm`, `kknn`, `ctree`, `cforest`, `lightgbm`, `xgboost`, `stacking`, `superlearner`; regression plus binary classification: `gam`, `bart`; classification only: `C50`, `naivebayes`, `fda`, `lda`, `qda`; binary classification only: `adaboost`; regression plus binary classification: `earth`; regression only: `pls`.

Usage

```
fit(
  formula,
  data,
  model,
  spec = NULL,
  seed = NULL,
  na_action = stats::na.fail,
  ...
)
```

Arguments

`formula`	Model formula.
`data`	Data frame.
`model`	Learner id (see `learners()`).
`spec`	Optional list of hyperparameters for the learner.
`seed`	Optional seed for reproducibility.
`na_action`	NA handling passed to `model.frame`/`model.matrix` (default `stats::na.fail`).
`...`	Additional parameters merged into `spec`.

Details

The learner engine packages are installed with `funcml`, so the advertised registry is intended to be available after a standard installation.

Value

An object of class `funcml_fit`.

Examples

```
fit_obj <- fit(mpg ~ wt + hp, data = mtcars, model = "glm")
predict(fit_obj, newdata = mtcars[1:3, , drop = FALSE])
```

`fit-methods` *Methods for fitted funcml models.*

Description

These methods provide the standard `print()`, `summary()`, `predict()`, and `coef()` interfaces for `funcml_fit` objects.

Usage

```
## S3 method for class 'funcml_fit'
print(x, ...)

## S3 method for class 'funcml_fit'
summary(object, ...)

## S3 method for class 'funcml_fit'
predict(
  object,
  newdata,
  type = NULL,
  class_level = NULL,
  pos_level = NULL,
  na_action = object$na_action,
  ...
)

## S3 method for class 'funcml_fit'
coef(object, ...)
```

Arguments

`x`	A `funcml_fit` object.
`...`	Additional arguments passed to the underlying method.
`object`	A `funcml_fit` object.
`newdata`	Data frame of new observations.
`type`	Prediction type override.
`class_level`	Target class for multiclass probability predictions.
`pos_level`	Alias for the binary positive class.
`na_action`	NA handling for new data.

Value

`print()` and `summary()` return the input object invisibly. `predict()` returns predictions in the requested format. `coef()` returns a named numeric coefficient vector when available.

Examples

```
fit_obj <- fit(mpg ~ wt + hp, data = mtcars, model = "glm")
print(fit_obj)
summary(fit_obj)
predict(fit_obj, newdata = mtcars[1:3, , drop = FALSE])
coef(fit_obj)
```

---

group_cv *Grouped cross-validation.*

Description

Grouped cross-validation.

Usage

```
group_cv(v = 5, group, repeats = 1, seed = NULL)
```

Arguments

`v`	Number of folds.
`group`	Grouping variable name or vector.
`repeats`	Number of repeats.
`seed`	Optional seed.

Value

A `funcml_cv` object.

Examples

```
group_cv(v = 3, group = rep(letters[1:3], each = 4), seed = 1)
```

haberman *Haberman survival data*

Description

A binary classification dataset on breast cancer survival after surgery.

Usage

```
haberman
```

Format

A data frame with 306 rows and 4 variables:

age Age of patient at operation time in years.

operation_year Year of operation minus 1900.

positive_axillary_nodes Number of positive axillary nodes detected.

survival_status Survival status (1 = survived 5 years or longer, 2 = died within 5 years).

Source

Haberman's Survival Data from the University of Chicago's Billings Hospital study, distributed through the UCI Machine Learning Repository.

Examples

```
str(funcml::haberman)
table(funcml::haberman$survival_status)
```

heartdisease *Heart disease patient data*

Description

A binary classification dataset on heart disease status using demographic and clinical risk factors.

Usage

```
heartdisease
```

Format

A data frame with 303 rows and 9 variables:

age Age in years.

sex Recorded sex.

chest_pain Chest pain type.

bp Resting blood pressure.

cholesterol Serum cholesterol measurement.

blood_sugar High fasting blood sugar indicator.

maximum_hr Maximum heart rate achieved.

exercise_induced_angina Exercise-induced angina indicator.

heart_disease Heart disease outcome (`"Yes"` or `"No"`).

Details

Column names were standardized to `snake_case` when packaging the data.

Source

CardioDataSets package dataset `CardioDataSets::heartdisease_tbl_df`.

Examples

```
str(funcml::heartdisease)
table(funcml::heartdisease$heart_disease)
```

---

heartfailure *Heart failure data*

Description

A binary classification dataset on heart failure mortality using demographic, laboratory, and clinical covariates.

Usage

```
heartfailure
```

Format

A data frame with 299 rows and 13 variables:

age Age in years.

anaemia Anaemia indicator.

creatinine_phosphokinase Creatinine phosphokinase level.

diabetes Diabetes indicator.

ejection_fraction Ejection fraction percentage.

high_blood_pressure High blood pressure indicator.

platelets Platelet count.

serum_creatinine Serum creatinine level.

serum_sodium Serum sodium level.

sex Sex indicator.

smoking Smoking indicator.

time Follow-up time.

death_event Death event outcome indicator (`0` = no event, `1` = death).

Details

Column names were standardized to `snake_case` when packaging the data.

Source

CardioDataSets package dataset `CardioDataSets::cardiac_failure_df`.

Examples

```
str(funcml::heartfailure)
table(funcml::heartfailure$death_event)
```

---

holdout *Plain holdout resampling.*

Description

Plain holdout resampling.

Usage

```
holdout(prop = 0.8, strata = TRUE, seed = NULL)
```

Arguments

prop	Training-set proportion.
strata	Logical; stratify classification outcomes.
seed	Optional seed.

Value

A funcml_cv object.

Examples

```
holdout(prop = 0.75, seed = 1)
```

---

infantmortality *Infant mortality data*

Description

A regression dataset on infant mortality with country-level income, region, and oil-export status covariates.

Usage

```
infantmortality
```

Format

A data frame with 105 rows and 5 variables:

country Country name.

income Per-capita income.

infant Infant mortality rate.

region Geographic region.

oil Oil-exporting country indicator.

Details

Column names were standardized to snake_case when packaging the data.

Source

Fox J, Weisberg S (2019). *An R Companion to Applied Regression*. Sage. The packaged data are from carData::Leinhardt.

Examples

```
str(funcml::infantmortality)
summary(funcml::infantmortality$infant)
```

interpret *Model-agnostic interpretation (global + local).*

Description

Implements native permutation VI, PDP/ICE/ALE, SHAP approximations, local surrogate explanations, interaction strength, and global surrogate models.

Usage

```
interpret(
  fit,
  data,
  formula = fit$formula,
  method = c("vip", "permute", "pdp", "ice", "ale", "local", "lime", "shap",
    "local_model", "interaction", "surrogate", "profile", "ceteris_paribus",
    "calibration"),
  features = NULL,
  type = NULL,
  metric = NULL,
  importance_type = c("permute", "model", "auto"),
  compare = c("difference", "ratio"),
  keep = TRUE,
  k = NULL,
  gower_power = NULL,
  class_level = NULL,
  pos_level = NULL,
  newdata = NULL,
  nsim = NULL,
  nsamples = NULL,
  grid = NULL,
  seed = NULL,
  bins = 10,
  strategy = c("quantile", "uniform"),
  ...
)
```

Arguments

fit	A funcml_fit object.
data	Reference data (typically training set).
formula	Optional formula (defaults to fit$formula).
method	One of "vip","permute","pdp","ice","ale","local","lime", "shap","local_model","interaction","surrogate","profile", "ceteris_paribus", or "calibration".
features	Optional subset of features; defaults to all predictors.

type	Prediction scale: regression -> "response"; classification -> "prob" or "class".
metric	Loss/score for importance (reg: rmse/mae/mse/medae/mape/rsq; cls: accuracy/precision/recall/specificity
importance_type	Importance engine for method = "vip". Internal permutation importance is always used; accepted values are retained only for backward-compatible argument parsing.
compare	How to compare baseline and perturbed performance for importance: "difference" or "ratio".
keep	Keep per-repetition raw importance scores when nsim > 1.
k	Sparsity target for local surrogate fits (method = "local", "local_model", or "lime").
gower_power	Exponent applied to native similarity weights when constructing the local neighborhood.
class_level	Target class for multiclass/local prob explanations.
pos_level	Alias for binary positive class (second level default).
newdata	Single-row data frame for local/SHAP explanations; defaults to first row of data.
nsim	Number of Monte Carlo simulations (importance/SHAP) or repetitions.
nsamples	Row subsample for speed (reference/background set).
grid	Optional list of grids per feature for PDP/ICE/ALE.
seed	Optional seed for determinism.
bins	Number of bins for calibration diagnostics.
strategy	Binning strategy for calibration diagnostics.
...	Additional method-specific args.

Value

An interpretation object whose class depends on method. Returned objects contain computed explanation values and metadata used for printing, summarizing, and plotting.

Examples

```
fit_obj <- fit(
  mpg ~ wt + hp + disp,
  data = mtcars,
  model = "rpart",
  spec = list(cp = 0.01, minsplit = 5)
)
vi <- interpret(
  fit = fit_obj,
  data = mtcars,
  method = "permute",
  features = c("wt", "hp"),
  nsim = 2,
  metric = "rmse"
)
vi$result$scores
```

interpret-ale-methods *ALE result methods.*

Description

These methods provide the standard `print()`, `summary()`, and `plot()` interfaces for `funcml_ale` objects returned by `interpret(method = "ale")`.

Usage

```
## S3 method for class 'funcml_ale'
plot(x, ...)

## S3 method for class 'funcml_ale'
print(x, ...)

## S3 method for class 'funcml_ale'
summary(object, ...)
```

Arguments

`x`	A `funcml_ale` object.
`...`	Additional arguments passed to the underlying method.
`object`	A `funcml_ale` object.

Value

`plot()` returns a `ggplot2` object. `print()` returns the input object invisibly. `summary()` returns the ALE curve table invisibly.

Examples

```
fit_obj <- fit(mpg ~ wt + hp + disp, data = mtcars, model = "rpart",
  spec = list(cp = 0.01, minsplit = 5))
ale <- interpret(
  fit = fit_obj,
  data = mtcars,
  method = "ale",
  features = c("wt", "hp"),
  nsamples = 20
)
print(ale)
summary(ale)
plot(ale)
```

`interpret-calibration-methods` *Calibration result methods.*

Description

These methods provide the standard `print()`, `summary()`, and `plot()` interfaces for `funcml_calibration` objects returned by `interpret(method = "calibration")`.

Usage

```
## S3 method for class 'funcml_calibration'
plot(x, style = c("curve", "histogram"), ...)

## S3 method for class 'funcml_calibration'
print(x, ...)

## S3 method for class 'funcml_calibration'
summary(object, ...)
```

Arguments

`x`	A `funcml_calibration` object.
`style`	Plot style: `"curve"` or `"histogram"`.
`...`	Additional arguments passed to the underlying method.
`object`	A `funcml_calibration` object.

Value

`plot()` returns a ggplot2 object. `print()` returns the input object invisibly. `summary()` returns a list with calibration curve and summary diagnostics invisibly.

Examples

```
fit_obj <- fit(mpg > 20 ~ wt + hp + disp, data = mtcars, model = "glm")
cal <- interpret(
  fit = fit_obj,
  data = mtcars,
  method = "calibration"
)
print(cal)
summary(cal)
plot(cal)
```

interpret-ice-methods *ICE result methods.*

Description

These methods provide the standard `print()`, `summary()`, and `plot()` interfaces for `funcml_ice` objects returned by `interpret(method = "ice")` or `interpret(method = "ceteris_paribus")`.

Usage

```
## S3 method for class 'funcml_ice'
plot(x, ...)

## S3 method for class 'funcml_ice'
print(x, ...)

## S3 method for class 'funcml_ice'
summary(object, ...)
```

Arguments

`x`	A `funcml_ice` object.
`...`	Additional arguments passed to the underlying method.
`object`	A `funcml_ice` object.

Value

`plot()` returns a `ggplot2` object. `print()` returns the input object invisibly. `summary()` returns the ICE curve table invisibly.

Examples

```
fit_obj <- fit(mpg ~ wt + hp + disp, data = mtcars, model = "rpart",
  spec = list(cp = 0.01, minsplit = 5))
ice <- interpret(
  fit = fit_obj,
  data = mtcars,
  method = "ice",
  features = c("wt", "hp"),
  nsamples = 20
)
print(ice)
summary(ice)
plot(ice)
```

interpret-interaction-methods
Interaction result methods.

Description

These methods provide the standard `print()`, `summary()`, and `plot()` interfaces for `funcml_interaction` objects returned by `interpret(method = "interaction")`.

Usage

```
## S3 method for class 'funcml_interaction'
plot(x, ...)

## S3 method for class 'funcml_interaction'
print(x, ...)

## S3 method for class 'funcml_interaction'
summary(object, ...)
```

Arguments

`x`	A `funcml_interaction` object.
`...`	Additional arguments passed to the underlying method.
`object`	A `funcml_interaction` object.

Value

`plot()` returns a `ggplot2` object. `print()` returns the input object invisibly. `summary()` returns the interaction summary table invisibly.

Examples

```
fit_obj <- fit(mpg ~ wt + hp + disp, data = mtcars, model = "rpart",
  spec = list(cp = 0.01, minsplit = 5))
interaction_obj <- interpret(
  fit = fit_obj,
  data = mtcars,
  method = "interaction",
  features = c("wt", "hp"),
  nsamples = 20,
  grid_size = 5
)
print(interaction_obj)
summary(interaction_obj)
plot(interaction_obj)
```

interpret-local-methods *Local surrogate result methods.*

Description

These methods provide the standard `print()`, `summary()`, and `plot()` interfaces for `funcml_local` objects returned by `interpret(method = "local")` or `interpret(method = "lime")`.

Usage

```
## S3 method for class 'funcml_local'
plot(x, ...)

## S3 method for class 'funcml_local'
print(x, ...)

## S3 method for class 'funcml_local'
summary(object, ...)
```

Arguments

`x`	A `funcml_local` object.
`...`	Additional arguments passed to the underlying method.
`object`	A `funcml_local` object.

Value

`plot()` returns a `ggplot2` object. `print()` returns the input object invisibly. `summary()` returns the local explanation payload invisibly.

Examples

```
fit_obj <- fit(mpg ~ wt + hp + disp, data = mtcars, model = "rpart",
  spec = list(cp = 0.01, minsplit = 5))
local_obj <- interpret(
  fit = fit_obj,
  data = mtcars,
  method = "local",
  newdata = mtcars[1, , drop = FALSE],
  k = 2
)
print(local_obj)
summary(local_obj)
plot(local_obj)
```

interpret-local-model-methods
Local surrogate result methods for `local_model` *and* `lime`.

Description

These methods provide the standard `print()`, `summary()`, and `plot()` interfaces for `funcml_iml_local_model` objects returned by `interpret(method = "local_model")` or `interpret(method = "lime")`.

Usage

```
## S3 method for class 'funcml_iml_local_model'
plot(x, ...)

## S3 method for class 'funcml_lime'
plot(x, ...)

## S3 method for class 'funcml_iml_local_model'
print(x, ...)

## S3 method for class 'funcml_iml_local_model'
summary(object, ...)
```

Arguments

`x`	A `funcml_iml_local_model` object.
`...`	Additional arguments passed to the underlying method.
`object`	A `funcml_iml_local_model` object.

Value

`plot()` returns a `ggplot2` object. `print()` returns the input object invisibly. `summary()` returns the local model explanation payload invisibly.

Examples

```
fit_obj <- fit(mpg ~ wt + hp + disp, data = mtcars, model = "rpart",
  spec = list(cp = 0.01, minsplit = 5))
local_model <- interpret(
  fit = fit_obj,
  data = mtcars,
  method = "local_model",
  newdata = mtcars[1, , drop = FALSE],
  k = 2
)
print(local_model)
summary(local_model)
plot(local_model)
```

`interpret-pdp-methods` *Partial dependence result methods.*

Description

These methods provide the standard `print()`, `summary()`, and `plot()` interfaces for `funcml_pdp` objects returned by `interpret(method = "pdp")`.

Usage

```
## S3 method for class 'funcml_pdp'
plot(x, ...)

## S3 method for class 'funcml_pdp'
print(x, ...)

## S3 method for class 'funcml_pdp'
summary(object, ...)
```

Arguments

`x`	A `funcml_pdp` object.
`...`	Additional arguments passed to the underlying method.
`object`	A `funcml_pdp` object.

Value

`plot()` returns a `ggplot2` object. `print()` returns the input object invisibly. `summary()` returns the partial dependence table invisibly.

Examples

```
fit_obj <- fit(mpg ~ wt + hp + disp, data = mtcars, model = "rpart",
  spec = list(cp = 0.01, minsplit = 5))
pdp <- interpret(
  fit = fit_obj,
  data = mtcars,
  method = "pdp",
  features = c("wt", "hp"),
  nsamples = 20
)
print(pdp)
summary(pdp)
plot(pdp)
```

`interpret-permute-methods` *Permutation importance result methods.*

Description

These methods provide the standard `print()`, `summary()`, and `plot()` interfaces for `funcml_permute` objects returned by `interpret(method = "permute")`.

Usage

```
## S3 method for class 'funcml_permute'
plot(x, ...)

## S3 method for class 'funcml_permute'
print(x, ...)

## S3 method for class 'funcml_permute'
summary(object, ...)
```

Arguments

`x`	A `funcml_permute` object.
`...`	Additional arguments passed to the underlying method.
`object`	A `funcml_permute` object.

Value

`plot()` returns a `ggplot2` object. `print()` returns the input object invisibly. `summary()` returns the permutation importance table invisibly.

Examples

```
fit_obj <- fit(mpg ~ wt + hp + disp, data = mtcars, model = "rpart",
  spec = list(cp = 0.01, minsplit = 5))
perm <- interpret(
  fit = fit_obj,
  data = mtcars,
  method = "permute",
  features = c("wt", "hp"),
  nsim = 1,
  metric = "rmse"
)
print(perm)
summary(perm)
plot(perm)
```

interpret-shap-methods *SHAP result methods.*

Description

These methods provide the standard `print()`, `summary()`, and `plot()` interfaces for `funcml_shap` objects returned by `interpret(method = "shap")`.

Usage

```
## S3 method for class 'funcml_shap'
plot(
  x,
 kind = c("auto", "waterfall", "force", "summary", "beeswarm", "importance", "bar",
    "dependence", "dependence2d", "interaction"),
  ...
)

## S3 method for class 'funcml_shap'
print(x, ...)

## S3 method for class 'funcml_shap'
summary(object, ...)
```

Arguments

`x`	A `funcml_shap` object.
`kind`	Plot kind. One of `"auto"`, `"waterfall"`, `"force"`, `"summary"`, `"beeswarm"`, `"importance"`, `"bar"`, `"dependence"`, `"dependence2d"`, or `"interaction"`.
`...`	Additional arguments passed to the underlying method.
`object`	A `funcml_shap` object.

Value

`plot()` returns a visualization object (typically a ggplot2 object) from `shapviz`. `print()` returns the input object invisibly. `summary()` returns the SHAP contribution table invisibly.

Examples

```
fit_obj <- fit(mpg ~ wt + hp + disp, data = mtcars, model = "rpart",
  spec = list(cp = 0.01, minsplit = 5))
shap <- interpret(
  fit = fit_obj,
  data = mtcars,
  method = "shap",
  newdata = mtcars[1, , drop = FALSE],
```

```
  nsim = 1
)
print(shap)
summary(shap)
plot(shap)
```

`interpret-surrogate-methods` *Global surrogate result methods.*

Description

These methods provide the standard `print()`, `summary()`, and `plot()` interfaces for `funcml_surrogate` objects returned by `interpret(method = "surrogate")`.

Usage

```
## S3 method for class 'funcml_surrogate'
plot(x, ...)

## S3 method for class 'funcml_surrogate'
print(x, ...)

## S3 method for class 'funcml_surrogate'
summary(object, ...)
```

Arguments

`x`	A `funcml_surrogate` object.
`...`	Additional arguments passed to the underlying method.
`object`	A `funcml_surrogate` object.

Value

`plot()` returns a `ggplot2` object. `print()` returns the input object invisibly. `summary()` returns the surrogate model summary object invisibly.

Examples

```
fit_obj <- fit(mpg ~ wt + hp + disp, data = mtcars, model = "rpart",
  spec = list(cp = 0.01, minsplit = 5))
surrogate <- interpret(
  fit = fit_obj,
  data = mtcars,
  method = "surrogate"
)
print(surrogate)
summary(surrogate)
plot(surrogate)
```

`ketapain` *Ketamine pain management data*

Description

A regression dataset on ketamine dosing, treatment characteristics, cost, quality-adjusted life years, and administration mode.

Usage

```
ketapain
```

Format

A data frame with 184 rows and 11 variables:

patient_id Patient identifier.

sexe Recorded sex.

age Patient age in years.

av_dose Average dose.

level_dose Dose level category.

cum_dose Cumulative dose.

cum_days Cumulative treatment days.

perfusion Perfusion duration.

cost Treatment cost.

qaly Quality-adjusted life years.

mode Administration mode.

Details

Column names were standardized to `snake_case` when packaging the data.

Source

Original ketamine pain management dataset distributed with the project materials.

Examples

```
str(funcml::ketapain)
summary(funcml::ketapain$qaly)
```

learners *Available learners.*

Description

`learners()` returns the registry keys accepted by `fit()`. Task support is: regression and classification: `glm`, `rpart`, `glmnet`, `ranger`, `nnet`, `e1071_svm`, `randomForest`, `gbm`, `kknn`, `ctree`, `cforest`, `lightgbm`, `xgboost`, `stacking`, `superlearner`; regression plus binary classification: `gam`, `bart`, `earth`; classification only: `C50`, `naivebayes`, `fda`, `lda`, `qda`; binary classification only: `adaboost`; regression only: `pls`.

Usage

```
learners()
```

Details

The learner engine packages are installed with `funcml`, so the advertised registry is intended to be available after a standard installation.

Value

Character vector of learner ids.

Examples

```
learners()
```

list_interpretability_methods *List available interpretability methods.*

Description

Returns a compact catalog of `interpret()` entry points and whether each method has a corresponding `plot()` method.

Usage

```
list_interpretability_methods(has_plot = NULL, columns = NULL)
```

Arguments

`has_plot`	Optional logical filter for methods with plot support.
`columns`	Optional character vector of columns to return.

Value

Data frame of interpretability methods.

Examples

```
list_interpretability_methods()
subset(list_interpretability_methods(), has_plot)
```

`list_learners` *Learner inventory table with capabilities.*

Description

`list_learners()` returns a compact learner registry in the style of a catalog table. By default it focuses on the most user-visible columns: learner id, generic fit/predict/tune entry points, and availability in the current session.

Usage

```
list_learners(
  has_fit = NULL,
  has_predict = NULL,
  has_tune = NULL,
  available = NULL,
  columns = NULL,
  regression = NULL,
  classification = NULL,
  prob = NULL,
  multiclass = NULL,
  importance = NULL,
  tune = NULL
)
```

Arguments

`has_fit`	Optional logical filter for fit support.
`has_predict`	Optional logical filter for predict support.
`has_tune`	Optional logical filter for tuning support.
`available`	Optional logical filter for engine availability in the current session.
`columns`	Optional character vector of columns to return.
`regression`	Optional logical filter for regression support.
`classification`	Optional logical filter for classification support.
`prob`	Optional logical filter for probability support.
`multiclass`	Optional logical filter for multiclass support.
`importance`	Optional logical filter for feature-importance support.
`tune`	Deprecated alias for `has_tune`.

Details

Additional capability metadata remains available through `columns =`.

Value

Data frame with learner metadata and capability columns.

Examples

```
list_learners()
list_learners(has_tune = TRUE)
list_tunable_learners()
list_learners(classification = TRUE, prob = TRUE,
              columns = c("learner", "has_tune", "supports_prob", "engine_package"))
```

---

list_metrics *List available metrics used in scoring and resampling summaries.*

Description

List available metrics used in scoring and resampling summaries.

Usage

```
list_metrics(direction = NULL, columns = NULL)
```

Arguments

`direction`	Optional character filter: `"maximize"` or `"minimize"`.
`columns`	Optional character vector of columns to return.

Value

Data frame of metric metadata.

Examples

```
list_metrics()
list_metrics(direction = "minimize")
```

list_tunable_learners *Shortcut for learners with tuning support.*

Description

Shortcut for learners with tuning support.

Usage

```
list_tunable_learners(...)
```

Arguments

...	Passed to `list_learners()`.

Value

Data frame with the same columns as `list_learners()`.

Examples

```
list_tunable_learners()
```

mammography *Mammography calcification data*

Description

A binary classification dataset for detection of mammographic microcalcifications.

Usage

```
mammography
```

Format

A data frame with 11,183 rows and 7 variables:

attr1 Numeric imaging-derived predictor 1.

attr2 Numeric imaging-derived predictor 2.

attr3 Numeric imaging-derived predictor 3.

attr4 Numeric imaging-derived predictor 4.

attr5 Numeric imaging-derived predictor 5.

attr6 Numeric imaging-derived predictor 6.

class Calcification class (\"-1\" or \"1\").

Source

Woods K, Doss C, Bowyer K, Solka J, Priebe C, Kegelmeyer P (1993). Comparative evaluation of pattern recognition techniques for detection of microcalcifications in mammography.

Examples

```
str(funcml::mammography)
table(funcml::mammography$class)
```

`metrics` *Regression and classification metrics.*

Description

Base R implementations used across evaluation and interpretation utilities.

Usage

```
rmse(truth, pred)

mae(truth, pred)

mse(truth, pred)

rsq(truth, pred)

medae(truth, pred)

mape(truth, pred)

logloss(truth, prob_matrix)

brier(truth, prob_matrix)

accuracy(truth, pred_class)

precision(truth, pred_class)

recall(truth, pred_class)

specificity(truth, pred_class)

f1(truth, pred_class)

balanced_accuracy(truth, pred_class)

auc(truth, prob, average = c("macro", "weighted"))
```

```
auc_weighted(truth, prob)

calibration_curve(
  truth,
  prob,
  bins = 10,
  strategy = c("quantile", "uniform"),
  positive = NULL
)

ece(
  truth,
  prob,
  bins = 10,
  strategy = c("quantile", "uniform"),
  positive = NULL
)

mce(
  truth,
  prob,
  bins = 10,
  strategy = c("quantile", "uniform"),
  positive = NULL
)
```

Arguments

`truth`	Observed outcomes.
`pred`	Predicted numeric values or class labels.
`prob_matrix`	Matrix or vector of predicted probabilities (classification).
`pred_class`	Predicted class labels (classification).
`prob`	Probability vector (binary) or probability matrix with one column per class (multiclass).
`average`	For multiclass AUC, aggregation mode: `"macro"` or `"weighted"` (class-frequency weighted one-vs-rest AUC). Ignored for binary AUC.
`bins`	Number of bins for calibration summaries.
`strategy`	Binning strategy: `"quantile"` or `"uniform"`.
`positive`	Optional positive/event class for binary classification.

Value

Numeric scalar metric.

Examples

```
truth_reg <- c(3, 5, 2.5, 7)
pred_reg <- c(2.8, 4.9, 2.7, 6.8)
rmse(truth_reg, pred_reg)
mae(truth_reg, pred_reg)
mse(truth_reg, pred_reg)
rsq(truth_reg, pred_reg)
medae(truth_reg, pred_reg)
mape(truth_reg, pred_reg)

truth_cls <- factor(c("no", "yes", "yes", "no"), levels = c("no", "yes"))
pred_cls <- factor(c("no", "yes", "no", "no"), levels = levels(truth_cls))
prob_cls <- cbind(
  no = c(0.8, 0.2, 0.6, 0.7),
  yes = c(0.2, 0.8, 0.4, 0.3)
)
logloss(truth_cls, prob_cls)
brier(truth_cls, prob_cls)
accuracy(truth_cls, pred_cls)
precision(truth_cls, pred_cls)
recall(truth_cls, pred_cls)
specificity(truth_cls, pred_cls)
f1(truth_cls, pred_cls)
balanced_accuracy(truth_cls, pred_cls)
auc(truth_cls, prob_cls[, "yes"])

truth_multi <- factor(c("a", "b", "c", "a", "b", "c"), levels = c("a", "b", "c"))
prob_multi <- rbind(
  c(0.90, 0.05, 0.05),
  c(0.05, 0.90, 0.05),
  c(0.05, 0.05, 0.90),
  c(0.85, 0.10, 0.05),
  c(0.10, 0.80, 0.10),
  c(0.05, 0.10, 0.85)
)
colnames(prob_multi) <- levels(truth_multi)
auc(truth_multi, prob_multi)
auc_weighted(truth_multi, prob_multi)
calibration_curve(truth_cls, prob_cls[, "yes"])
ece(truth_cls, prob_cls[, "yes"])
mce(truth_cls, prob_cls[, "yes"])
```

---

newthyroid *Thyroid function data*

Description

A multiclass classification dataset on thyroid functional state using five laboratory test measurements.

Usage

```
newthyroid
```

Format

A data frame with 215 rows and 6 variables:

t3_resin_uptake T3-resin uptake percentage.

total_serum_thyroxin Total serum thyroxin measurement.

total_serum_triiodothyronine Total serum triiodothyronine measurement.

basal_tsh Basal thyroid-stimulating hormone measurement.

max_abs_tsh_diff Maximum absolute TSH difference after thyrotropin- releasing hormone injection.

class Thyroid class (1 = normal, 2 = hyperthyroid, 3 = hypothyroid).

Source

Thyroid gland data distributed through the UCI Machine Learning Repository.

Examples

```
str(funcml::newthyroid)
table(funcml::newthyroid$class)
```

---

pimadiabetes *Pima diabetes data*

Description

A diabetes classification dataset with clinical measurements and a predefined train/test split column.

Usage

```
pimadiabetes
```

Format

A data frame with 532 rows and 9 variables:

npreg Number of pregnancies.

glu Plasma glucose concentration.

bp Diastolic blood pressure.

skin Triceps skin fold thickness.

bmi Body mass index.

ped Diabetes pedigree function.

age Age in years.

diabetes Diabetes outcome (`"Yes"` or `"No"`).

split Suggested split indicator (`"train"` or `"test"`).

Source

National Institute of Diabetes and Digestive and Kidney Diseases Pima Indians Diabetes Database.

References

Smith JW, Everhart JE, Dickson WC, Knowler WC, Johannes RS (1988). Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In *Proceedings of the Annual Symposium on Computer Application in Medical Care*, 261-265.

Examples

```
str(funcml::pimadiabetes)
table(funcml::pimadiabetes$split)
```

---

theme_funcml *FuncML plotting theme.*

Description

A custom ggplot2 theme used across `funcml` plots. It keeps a clean light background, restrained grid lines, and high-contrast labels so package figures remain consistent and publication-friendly.

Usage

```
theme_funcml(base_size = 11)
```

Arguments

`base_size`	Base text size passed to the theme.

Value

A ggplot2 theme object.

Examples

```
ggplot2::ggplot(mtcars, ggplot2::aes(wt, mpg)) +
  ggplot2::geom_point() +
  theme_funcml()
```

time_cv *Time-aware rolling resampling.*

Description

Time-aware rolling resampling.

Usage

```
time_cv(
  initial,
  assess = 1,
  time = NULL,
  skip = 0,
  cumulative = TRUE,
  seed = NULL
)
```

Arguments

initial	Initial training window size.
assess	Assessment window size.
time	Ordering variable name or vector.
skip	Number of observations to skip between splits.
cumulative	Logical; use an expanding training window.
seed	Optional seed.

Value

A funcml_cv object.

Examples

```
time_cv(initial = 8, assess = 2, skip = 1)
```

tune *Hyperparameter tuning via grid or random search.*

Description

Hyperparameter tuning via grid or random search.

Usage

```
tune(
  data,
  formula,
  model,
  grid,
  resampling = cv(5),
  metric = NULL,
  type = NULL,
  search = c("grid", "random"),
  n_evals = NULL,
  outer_resampling = NULL,
  seed = NULL,
  ncores = NULL,
  ...
)
```

Arguments

`data`	Data frame.
`formula`	Model formula.
`model`	Learner id.
`grid`	Data frame of hyperparameter combinations.
`resampling`	Resampling object.
`metric`	Metric to optimize.
`type`	Prediction type override.
`search`	Search strategy: `"grid"` or `"random"`.
`n_evals`	Maximum number of configurations to evaluate when `search = "random"`.
`outer_resampling`	Optional outer resampling object. When supplied, `tune()` performs nested resampling and reports outer-fold performance estimates for the tuned model-selection procedure.
`seed`	Optional seed.
`ncores`	Optional number of CPU cores used for tuning tasks. `NULL` or `1` runs sequentially.
`...`	Passed to `fit()`.

Value

A `funcml_tune` object.

Examples

```
tune_obj <- tune(
  data = mtcars,
  formula = mpg ~ wt + hp,
  model = "rpart",
  grid = expand.grid(cp = c(0.001, 0.01), minsplit = c(5, 10)),
  resampling = cv(3, seed = 1),
  metric = "rmse"
)
tune_obj$best
```

---

tune-methods *Methods for tuning results.*

Description

These methods provide the standard `print()`, `summary()`, and `plot()` interfaces for `funcml_tune` objects.

Usage

```
## S3 method for class 'funcml_tune'
print(x, ...)

## S3 method for class 'funcml_tune'
summary(object, ...)

## S3 method for class 'funcml_tune'
plot(x, ...)
```

Arguments

`x`	A `funcml_tune` object.
`...`	Additional arguments passed to the underlying method.
`object`	A `funcml_tune` object.

Value

`print()` and `summary()` return the input object or results table invisibly. `plot()` returns a `ggplot2` object.

Examples

```
tune_obj <- tune(
  data = mtcars,
  formula = mpg ~ wt + hp,
  model = "rpart",
  grid = expand.grid(cp = c(0.001, 0.01), minsplit = c(5, 10)),
```

```
  resampling = cv(3, seed = 1),
  metric = "rmse"
)
print(tune_obj)
summary(tune_obj)
plot(tune_obj)
```

Index