Package: PEAXAI (via r-universe)

November 17, 2026

Title Probabilistic Efficiency Analysis Using Explainable Artificial Intelligence

Version 1.0.3

Description Provides a probabilistic framework that integrates Data Envelopment Analysis (DEA) (Banker et al., 1984) <doi:10.1287/mnsc.30.9.1078> with machine learning classifiers (Kuhn, 2008) <doi:10.18637/jss.v028.i05> to estimate both the (in)efficiency status and the probability of efficiency for decision-making units. The approach trains predictive models on DEA-derived efficiency labels (Charnes et al., 1985) <doi:10.1016/0304-4076(85)90133-2>, enabling explainable artificial intelligence (XAI) workflows with global and local interpretability tools, including permutation importance (Molnar et al., 2018) <doi:10.21105/joss.00786>, Shapley value explanations (Strumbelj & Kononenko, 2014) <doi:10.1007/s10115-013-0679-x>, and sensitivity analysis (Cortez, 2011) <https://CRAN.R-project.org/package=rminer>. The framework also supports probability-threshold peer selection and counterfactual improvement recommendations for benchmarking and policy evaluation. The probabilistic efficiency framework is detailed in González-Moyano et al. (2025) ``Probability-based Technical Efficiency Analysis through Machine Learning", in review for publication.

License GPL-3

URL https://github.com/rgonzalezmoyano/PEAXAI

BugReports https://github.com/rgonzalezmoyano/PEAXAI/issues

Encoding UTF-8

Language en

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Depends R (>= 4.4.0)

Imports Benchmarking, caret, deaR, dplyr, kernelshap, iml, lime, np, PRROC, pROC, rminer, rms, stats, peakRAM

Suggests ggplot2, knitr, rmarkdown

VignetteBuilder knitr

LazyData false

ByteCompile true

Config/testthat/edition 3

Config/pak/sysreqs cmake libfreetype6-dev libglpk-dev libglu1-mesa-dev make texlive libicu-dev libpng-dev libuv1-dev libxml2-dev libgl1-mesa-dev libssl-dev zlib1g-dev

Repository https://rgonzalezmoyano.r-universe.dev

Date/Publication 2026-09-14 18:01:18 UTC

RemoteUrl https://github.com/rgonzalezmoyano/peaxai

RemoteRef HEAD

RemoteSha b271c0a07c32866ad077c37cb8ab9847eb758bed

Contents

convex_facets . . . 2
data . . . 3
data_example . . . 4
data_SABI . . . 5
firms . . . 7
get_SMOTE_DMUs . . . 8
PEAXAI_counterfactuals . . . 10
PEAXAI_fitting . . . 13
PEAXAI_global_importance . . . 15
PEAXAI_local_importance . . . 18
PEAXAI_peer . . . 21
PEAXAI_predict . . . 24
PEAXAI_ranking . . . 25
preprocessing . . . 28
reffcy . . . 29
train_PEAXAI . . . 30
xai_prepare_sets . . . 31

Index **33**

`convex_facets` *Identify Maximal Facets of the Convex Frontier*

Description

Identifies the maximal subsets of efficient Decision-Making Units (DMUs) that lie on the same face (facet) of the efficient frontier. It relies on the concept of "maximal friends" to determine which efficient units can be grouped together to form convex combinations under the chosen returns-to-scale assumption.

Usage

```
convex_facets(data, x, y, RTS = "vrs", verbose)
```

Arguments

data	A data.frame or matrix containing the variables used in the model.
x	Integer vector indicating the column indexes of the input variables in data.
y	Integer vector indicating the column indexes of the output variables in data.
RTS	Text string or number defining the underlying DEA technology / returns-to-scale assumption (default: "vrs"). Accepted values: 1 / "vrs" Variable returns to scale, convexity and free disposability. 3 / "crs" Constant returns to scale, convexity and free disposability.
verbose	Logical; if TRUE, prints progress messages (default FALSE).

Value

A matrix where each row represents a maximal facet, and the values are the row indices of the efficient DMUs in data that form that facet. Returns an empty matrix if there are no efficient units.

---

data *Simulated efficiency dataset (100 DMUs)*

Description

Dataset with 100 simulated decision-making units (DMUs) used to illustrate the basic workflow of **PEAXAI** in a simple single-input/single-output setting.

Usage

```
data(data)
```

Format

A data.frame with 100 rows and 3 columns:

x1 Input of the DMU (e.g., resource use, cost or effort).

y Observed output, potentially affected by technical inefficiency.

yD Deterministic (theoretical) output on the efficient frontier.

Details

Each DMU uses one input x1 to produce an output y. The variable yD represents the theoretical output on the deterministic frontier, that is, the output level that would be observed in the absence of technical inefficiency.

The dataset is purely simulated and is intended for examples and vignettes. It contains 100 DMUs with heterogeneous input levels and corresponding output levels. The observed output y can be interpreted as y <= yD, where the gap between yD and y reflects technical inefficiency (plus possible noise, depending on how the data were generated).

Source

Simulated data generated by the authors for illustrative purposes.

Examples

```
data(data)
str(data)
summary(data)

if (requireNamespace("ggplot2", quietly = TRUE)) {
  ggplot2::ggplot(data, ggplot2::aes(x = x1)) +
    ggplot2::geom_point(ggplot2::aes(y = y), alpha = 0.6) +
    ggplot2::geom_line(ggplot2::aes(y = yD), color = "red") +
    ggplot2::labs(
      x = "Input x1",
      y = "Output",
      title = "Simulated DMUs and theoretical frontier"
    ) +
    ggplot2::theme_minimal()
}
```

---

data_example *Simulated efficiency dataset (100 DMUs)*

Description

Dataset with 100 simulated decision-making units (DMUs) used to illustrate the basic workflow of **PEAXAI** in a simple single-input/single-output setting.

Usage

```
data(data_example)
```

Format

A `data.frame` with 100 rows and 3 columns:

x1 Input of the DMU (e.g., resource use, cost or effort).

y Observed output, potentially affected by technical inefficiency.

yD Deterministic (theoretical) output on the efficient frontier.

Details

Each DMU uses one input `x1` to produce an output `y`. The variable `yD` represents the theoretical output on the deterministic frontier, that is, the output level that would be observed in the absence of technical inefficiency.

The dataset is purely simulated and is intended for examples and vignettes. It contains 100 DMUs with heterogeneous input levels and corresponding output levels. The observed output `y` can be interpreted as `y <= yD`, where the gap between `yD` and `y` reflects technical inefficiency (plus possible noise, depending on how the data were generated).

Source

Simulated data generated by the authors for illustrative purposes.

Examples

```
data(data_example)
str(data_example)
summary(data_example)

if (requireNamespace("ggplot2", quietly = TRUE)) {
  ggplot2::ggplot(data_example, ggplot2::aes(x = x1)) +
    ggplot2::geom_point(ggplot2::aes(y = y), alpha = 0.6) +
    ggplot2::geom_line(ggplot2::aes(y = yD), color = "red") +
    ggplot2::labs(
      x = "Input x1",
      y = "Output",
      title = "Simulated DMUs and theoretical frontier"
    ) +
    ggplot2::theme_minimal()
}
```

---

data_SABI *Spanish Food Industry Firms Dataset*

Description

Dataset containing information on food industry companies located in Spain, used to illustrate efficiency analysis within the **PEAXAI** package.

Usage

```
data(data_SABI)
```

Format

A `data.frame` with 1267 rows and 4 columns:

fixed_assets Tangible fixed assets (euros).

personnel_expenses Personnel expenses (euros).

operating_income Operating income (euros).

firm_id Anonymized firm identifier identifying each DMU.

Details

The dataset includes 1267 food industry firms, collected from the *SABI* database for the year 2024. Each observation corresponds to a single company. Variables reflect both operational and financial dimensions relevant for productivity and efficiency assessment.

The output variable is:

- `operating_income` — Operating income (in euros), measuring revenues generated from core business activities.

The input variables are:

- `fixed_assets` — Tangible fixed assets (euros), such as buildings and machinery.
- `personnel_expenses` — Personnel expenses (millions of euros), including salaries, benefits, and training.

The sample displays substantial dispersion across variables, encompassing both small and large firms. This heterogeneity affects measures of central tendency—mean and median values differ considerably—thus providing a realistic challenge for efficiency and explainability analyses.

Source

SABI (Sistema de Análisis de Balances Ibéricos) database, 2024.

Examples

```
data(data_SABI)
str(data_SABI)
summary(data_SABI)

if (requireNamespace("ggplot2", quietly = TRUE)) {
 ggplot2::ggplot(data_SABI, ggplot2::aes(x = personnel_expenses, y = operating_income)) +
   ggplot2::geom_point(alpha = 0.6) +
   ggplot2::labs(
     x = "Personnel expenses",
     y = "Operating income (euros)",
     title = "Spanish Food Industry Firms (2024)"
   ) +
```

```
    ggplot2::theme_minimal() +
    ggplot2::theme(
      plot.title = ggplot2::element_text(face = "bold"),
      axis.line = ggplot2::element_line(color = "black"),
      axis.ticks = ggplot2::element_line(color = "black"),
      panel.grid.minor = ggplot2::element_blank()
    )
}
```

`firms` *Spanish Food Industry Firms Dataset*

Description

Dataset containing information on food industry companies located in Spain, used to illustrate efficiency analysis within the **PEAXAI** package. The dataset reflects the institutional and market heterogeneity that shapes firm-level efficiency across Spain's 17 autonomous communities.

Usage

```
data(firms)
```

Format

A `data.frame` with 917 rows and 6 columns:

total_assets Total assets (millions of euros).

employees Number of employees.

fixed_assets Tangible fixed assets (millions of euros).

personnel_expenses Personnel expenses (millions of euros).

operating_income Operating income (millions of euros).

autonomous_community Autonomous community where the firm operates.

Details

The dataset includes 917 food industry firms with more than 50 employees, collected from the *SABI* database for the year 2023. Each observation corresponds to a single company. Variables reflect both operational and financial dimensions relevant for productivity and efficiency assessment.

The output variable is:

- `operating_income` — Operating income (in millions of euros), measuring revenues generated from core business activities.

The input variables are:

- `total_assets` — Total assets (millions of euros), representing resources employed.

- `employees` — Number of employees, indicating workforce size (only firms with more than 50 workers are included).
- `fixed_assets` — Tangible fixed assets (millions of euros), such as buildings and machinery.
- `personnel_expenses` — Personnel expenses (millions of euros), including salaries, benefits, and training.

The variable `autonomous_community` identifies the territorial location of each firm within Spain.

The sample displays substantial dispersion across variables, encompassing both small and large firms. This heterogeneity affects measures of central tendency—mean and median values differ considerably—thus providing a realistic challenge for efficiency and explainability analyses.

Source

SABI (Sistema de Análisis de Balances Ibéricos) database, 2023. Firms with more than 50 employees in the Spanish food industry.

Examples

```
data(firms)
str(firms)
summary(firms)

if (requireNamespace("ggplot2", quietly = TRUE)) {
  ggplot2::ggplot(firms, ggplot2::aes(x = employees, y = operating_income)) +
    ggplot2::geom_point(alpha = 0.6) +
    ggplot2::labs(
      x = "Number of employees",
      y = "Operating income (millions of euros)",
      title = "Spanish Food Industry Firms (2023)"
    ) +
    ggplot2::theme_minimal() +
    ggplot2::theme(
      plot.title = ggplot2::element_text(face = "bold"),
      axis.line = ggplot2::element_line(color = "black"),
      axis.ticks = ggplot2::element_line(color = "black"),
      panel.grid.minor = ggplot2::element_blank()
    )
}
```

---

get_SMOTE_DMUs *Create New SMOTE Units to Balance Data combinations of m + s*

Description

This function creates new DMUs to address data imbalances. If the majority class is efficient, it generates new inefficient DMUs by worsering the observed units. Conversely, if the majority class is inefficient, it projects inefficient DMUs to the frontier. Finally, a random selection if performed to keep a proportion of 0.65 for the majority class and 0.35 for the minority class.

Usage

```
get_SMOTE_DMUs(
  data,
  facets = NULL,
  x,
  y,
  RTS = "vrs",
  balance_data = NULL,
  SMOTE_DDF = FALSE,
  z_numeric = NULL,
  z_factor = NULL,
  alpha = FALSE,
  bandwidth = NULL,
  seed
)
```

Arguments

data	A list of data.frames, where each element represents a dataset with labeled data.
facets	A list where each element represents a subgroup containing index combinations that generate efficient units.
x	Column indexes of the input variables in the data.
y	Column indexes of the output variables in the data.
RTS	Text string or number defining the underlying DEA technology / returns-to-scale assumption (default: "vrs"). Accepted values: 0 / "fdh" Free disposability hull, no convexity assumption. 1 / "vrs" Variable returns to scale, convexity and free disposability. 2 / "drs" Decreasing returns to scale, convexity, down-scaling and free disposability. 3 / "crs" Constant returns to scale, convexity and free disposability. 4 / "irs" Increasing returns to scale (up-scaling, not down-scaling), convexity and free disposability. 5 / "add" Additivity (scaling up and down, but only with integers), and free disposability.
balance_data	A numeric vector indicating the different levels of balance required (e.g., c(0.1, 0.45, 0.6)).
SMOTE_DDF	Logical. Determines how synthetic efficient units are generated. By default, FALSE, units are generated within full-dimensional facets of the DEA technology. If TRUE, a computationally simpler alternative based on DDF projections with randomly generated directional vectors is used.
z_numeric	Column indexes of the continuous environment variables (z) in the data.
z_factor	Column indexes of the factor environment variables (z) in the data.
alpha	This allow to choose the size of the Confidence Intervals computed. By defaulta alpha = FALSE. In this case no confidence interval are computed.

bandwidth	the bandwidth parameters for the unconditional kernel density estimator used in the conditional DEA framework. It is typically obtained using npudensbw and supports mixed data types, including continuous variables and discrete unordered or ordered factors. Bandwidths can be selected using normal reference rules, likelihood cross-validation, or least-squares cross-validation following Li and Racine (2003). If NULL, the bandwidth is estimated internally.
seed	Integer. Seed for reproducibility.

Value

A list where each element corresponds to a balance level, containing a single data.frame with the real and synthetic DMUs, correctly labeled.

PEAXAI_counterfactuals

Projection-Based Efficiency counterfactuals

Description

Computes efficiency projections for each observation based on a trained classifier from **caret** that provides class probabilities via predict(type = "prob"). For each probability threshold, the function finds the direction and magnitude of change in input–output space required for a unit to reach a specified efficiency level, following a directional distance approach.

Usage

```
PEAXAI_counterfactuals(
  data,
  x,
  y,
  z_numeric = NULL,
  z_factor = NULL,
  final_model,
  efficiency_thresholds,
  directional_vector,
  n_expand,
  n_grid,
  max_y = 1,
  min_x = 1
)
```

Arguments

data	A data.frame or matrix containing input and output variables.
x	A numeric vector indicating the column indexes of input variables in data.
y	A numeric vector indicating the column indexes of output variables in data.

`z_numeric` Integer vector with column indices of numeric environment variables in `data`. By default is `NULL`.

`z_factor` Integer vector with column indices of factor environment variables in `data`. By default is `NULL`.

`final_model` A fitted **caret** model of class `"train"` that supports `predict(type = "prob")` and returns a probability column for the efficient class.

`efficiency_thresholds` A numeric vector of probability levels in (0,1) that define the efficiency classes (e.g., `c(0.75, 0.9, 0.95)`).

`directional_vector` A `list` with the required information to construct the directional vector, including:

- `relative_importance`: Numeric vector of variable importances that sum to 1.
- `baseline`: `"mean"`, `"median"`, `"self"` or `"ones"`.

`n_expand` Numeric. Number of expansion steps used to enlarge the initial search range for β.

`n_grid` Integer. Number of grid points evaluated during each iteration to refine the cutoff value of β.

`max_y` Numeric. Upper-limit multiplier for output expansion in the search procedure (default = 1).

`min_x` Numeric. Lower-limit multiplier for input contraction in the search procedure (default = 1).

Details

For each observation and for each probability level in `efficiency_thresholds`, the function searches for the smallest directional distance β such that the predicted probability of belonging to the efficient class reaches the target.

Value

A named `list` with one element per threshold (names are the thresholds as character strings). Each element contains:

- `counterfactual_dataset`: A `data.frame` of projected input-output values at that threshold.
- `inefficiencies`: A `data.frame` with the optimal β and the corresponding predicted probability.

See Also

`find_beta_maxmin` for initializing search bounds; `train` for model training.

Examples

```
data("firms", package = "PEAXAI")

# Firms of the Valencian Community: four inputs and one output
data <- subset(
  firms,
  autonomous_community == "Comunidad Valenciana"
)

x <- 1:4
y <- 5

models <- PEAXAI_fitting(
  data            = data,
  x               = x,
  y               = y,
  RTS             = "vrs",
  imbalance_rate  = NULL,
  methods         = list("glm" = list(weights = "dinamic")),
  trControl       = list(method = "cv", number = 3),
  metric_priority = c("Balanced_Accuracy", "F1", "ROC_AUC"),
  seed            = 1,
  verbose         = FALSE
)

final_model <- models[["best_model_fit"]][["glm"]]

# The direction of the counterfactual search is given by the global
# importance of the variables, rescaled around the sample mean.
relative_importance <- PEAXAI_global_importance(
  final_model       = final_model,
  x                 = x,
  y                 = y,
  explain_data      = data,
  reference_data    = final_model$trainingData,
  importance_method = list(name = "PI", n.repetitions = 5),
  seed              = 1
)

directional_vector <- list(
  relative_importance = relative_importance,
  baseline            = "mean"
)

efficiency_thresholds <- c(0.75, 0.90)

targets <- PEAXAI_counterfactuals(
  data                  = data,
  x                     = x,
  y                     = y,
  final_model           = final_model,
  efficiency_thresholds = efficiency_thresholds,
```

```
  directional_vector    = directional_vector,
  n_expand              = 0.5,
  n_grid                = 50,
  max_y                 = 2,
  min_x                 = 1
)

# One element per threshold, each with the projected DMUs and their beta
names(targets)
head(targets[["0.75"]][["counterfactual_dataset"]])
head(targets[["0.75"]][["inefficiencies"]])
```

PEAXAI_fitting *Training Classification Models to Estimate Efficiency*

Description

Trains one or multiple classification algorithms to identify Pareto-efficient decision-making units (DMUs). It jointly searches model hyperparameters and the class-balancing level (synthetic samples via SMOTE) using k-fold cross- validation or a train/validation/test split, selecting the configuration that maximizes the specified metric(s). Returns, for each technique, the best fitted model together with training summaries, performance metrics, and the selected balancing level.

Usage

```
PEAXAI_fitting(
  data,
  x,
  y,
  RTS = "vrs",
  imbalance_rate = NULL,
  trControl,
  methods,
  metric_priority = "Balanced_Accuracy",
  hold_out = NULL,
  seed = 314,
  verbose = TRUE,
  z_numeric = NULL,
  z_factor = NULL,
  B = NULL,
  alpha = NULL,
  m = NULL
)
```

Arguments

data	A data.frame or matrix containing the variables in the model.
x	Integer vector with column indices of input variables in data.
y	Integer vector with column indices of output variables in data.
RTS	Text string or number defining the underlying DEA technology / returns-to-scale assumption (default: "vrs"). Accepted values: 1 / "vrs" Variable returns to scale, convexity and free disposability. 3 / "crs" Constant returns to scale, convexity and free disposability.
imbalance_rate	Optional target(s) for class balance via SMOTE. If NULL, no synthetic balancing is performed.
trControl	A caret::trainControl-like list that specifies the resampling strategy; recognized values for $method include "cv", "test_set", and "none". See **caret** documentation.
methods	A list of selected machine learning models and their hyperparameters.
metric_priority	A string specifying the summary metric for classification to select the optimal model. Default includes "Balanced_Accuracy" due to (normally) unbalanced data. Calibration is performed using validation data whenever resampling is enabled.
hold_out	Numeric proportion in (0,1) for validation split (default NULL). If NULL, training and validation use the same data.
seed	Integer. Seed for reproducibility.
verbose	Logical; if TRUE, prints progress messages (default FALSE).
z_numeric	Integer vector with column indices of numeric environment variables in data. By default is NULL.
z_factor	Integer vector with column indices of factor environment variables in data. By default is NULL.
B	number of bootstrap replicates in Conditional DEA.
alpha	This allow to choose the size of the Confidence Intervals computed. By defaulta alpha = FALSE. In this case no confidence interval are computed.
m	number of units to be included in the reference set.

Value

A "PEAXAI" (list) with the best technique, best fitted models and their performance and the results by fold.

Examples

```
data("firms", package = "PEAXAI")

# Firms of the Valencian Community: 97 DMUs described by four inputs
# (columns 1 to 4) and one output (column 5).
data <- subset(
```

```
  firms,
  autonomous_community == "Comunidad Valenciana"
)

x <- 1:4
y <- 5

# Resampling scheme: stratified 3-fold cross-validation
trControl <- list(
  method = "cv",
  number = 3
)

# Candidate algorithm: logistic regression with dynamic class weights
methods <- list(
  "glm" = list(
    weights = "dinamic"
  )
)

models <- PEAXAI_fitting(
  data            = data,
  x               = x,
  y               = y,
  RTS             = "vrs",
  imbalance_rate  = NULL,
  methods         = methods,
  trControl       = trControl,
  metric_priority = c("Balanced_Accuracy", "F1", "ROC_AUC"),
  seed            = 1,
  verbose         = FALSE
)

# Best fitted model for the 'glm' technique
final_model <- models[["best_model_fit"]][["glm"]]
```

`PEAXAI_global_importance` *Global feature importance for efficiency classifiers*

Description

Computes **global feature importance** for a fitted classification model that separates Pareto-efficient DMUs, using one of three XAI backends:

- `"SA"` — Sensitivity Analysis via **rminer**.
- `"SHAP"` — Model-agnostic SHAP approximations via **kernelshap**.
- `"PI"` — Permutation Importance via **iml**.

Importance is computed on the DMUs supplied in `explain_data`, using `reference_data` (typically `final_model$trainingData`) as the reference distribution. Importances are returned normalized to sum to 1.

Usage

```
PEAXAI_global_importance(
  final_model,
  x,
  y,
  z_numeric = NULL,
  z_factor = NULL,
  explain_data,
  reference_data,
  importance_method,
  seed = 314
)
```

Arguments

`final_model`	A fitted model. If it is a base-`glm` binomial, probabilities are obtained with `type = "response"`; otherwise the function expects `predict(type = "prob")` with a column named `"efficient"`.
`x`	Integer or character vector with the columns used as **inputs** (predictors).
`y`	Integer or character vector with the columns used as **outputs** (targets used to define `class_efficiency` in training; not included in `X` when explaining).
`z_numeric`	Integer vector with column indices of numeric environment variables in `data`. By default is `NULL`.
`z_factor`	Integer vector with column indices of factor environment variables in `data`. By default is `NULL`.
`explain_data`	A `data.frame` with the original observed DMUs passed by the user to `PEAXAI_fitting()`. This dataset represents the real, non-augmented observations on which explanations can be computed.
`reference_data`	A `data.frame` with the training dataset used to fit the optimal model returned by `PEAXAI_fitting()`. Depending on the class- balancing procedure, this dataset may contain only real observed DMUs or both real and synthetic DMUs.
`importance_method`	A named list (or data.frame-like) with the backend and its args: `name` One of `"SA"`, `"SHAP"`, `"PI"`. `method` (SA) One of `"1D-SA"`, `"sens"`, `"DSA"`, `"MSA"`, `"CSA"`, `"GSA"`. `measures` (SA) e.g. `"AAD"`, `"gradient"`, `"variance"`, `"range"`. `levels` (SA) Discretization levels used by `rminer::Importance`. `baseline` (SA) Baseline value for SA, if applicable. `bg_n` (SHAP) Background sample size for `kernelshap::kernelshap` (default: 200). `n.repetitions` (PI) Number of permutations per feature for `iml::FeatureImp`.
`seed`	Integer. Seed for reproducibility.

Details

Internally, the function builds background/target sets with `xai_prepare_sets()`. For `glm` models, the positive class is assumed to be the **second level** (`"efficient"`) and probabilities are extracted with `type = "response"`. For other models (e.g., **caret**), `predict(type = "prob")[, "efficient"]` is used.

Value

A named numeric vector (or 1-row data.frame) of normalized importances, with names matching the predictor columns; the values sum to 1.

See Also

kernelshap, FeatureImp, Importance

Examples

```
data("firms", package = "PEAXAI")

# Firms of the Valencian Community: four inputs and one output
data <- subset(
  firms,
  autonomous_community == "Comunidad Valenciana"
)

x <- 1:4
y <- 5

models <- PEAXAI_fitting(
  data            = data,
  x               = x,
  y               = y,
  RTS             = "vrs",
  imbalance_rate  = NULL,
  methods         = list("glm" = list(weights = "dinamic")),
  trControl       = list(method = "cv", number = 3),
  metric_priority = c("Balanced_Accuracy", "ROC_AUC"),
  seed            = 1,
  verbose         = FALSE
)

final_model <- models[["best_model_fit"]][["glm"]]

# Permutation importance (iml). Use list(name = "SHAP", bg_n = 50) for
# kernelshap, or list(name = "SA", method = "1D-SA") for rminer.
importance_method <- list(name = "PI", n.repetitions = 5)

relative_importance <- PEAXAI_global_importance(
  final_model       = final_model,
  x                 = x,
  y                 = y,
```

```
  explain_data      = data,
  reference_data    = final_model$trainingData,
  importance_method = importance_method,
  seed              = 1
)

# One normalized weight per input and output; they add up to 1
relative_importance
```

`PEAXAI_local_importance` *Local feature importance for efficiency classifiers*

Description

Computes **local feature importance** (i.e., per-observation explanations) for a fitted classification model that separates Pareto-efficient DMUs, using one of three XAI backends:

- `"SA"` — Local Sensitivity Analysis via **rminer**.
- `"SHAP"` — Model-agnostic SHAP approximations via **kernelshap**.
- `"LIME"` — Model-agnostic LIME approximations via **lime**

You can compute local importance on specific DMUs provided in `explain_data`, using `reference_data` as the reference distribution or training background. For each target DMU, importances are returned normalized to sum to 1 **within that DMU**.

Usage

```
PEAXAI_local_importance(
  final_model,
  x,
  y,
  z_numeric = NULL,
  z_factor = NULL,
  explain_data,
  reference_data,
  importance_method,
  seed = 314
)
```

Arguments

`final_model`	A fitted model. If it is a base-`glm` binomial, probabilities are obtained with `type = "response"`; otherwise the function expects `predict(type = "prob")` with a column named `"efficient"`.
`x`	Integer or character vector with the columns used as **inputs** (predictors).

y — Integer or character vector with the columns used as **outputs** (targets used to define `class_efficiency` in training; not included in `X` when explaining).

z_numeric — Integer vector with column indices of numeric environment variables in `data`. By default is `NULL`.

z_factor — Integer vector with column indices of factor environment variables in `data`. By default is `NULL`.

explain_data — A `data.frame` with the original observed DMUs (or new DMUs) on which **local** explanations are computed.

reference_data — A `data.frame` with the training dataset used to fit the optimal model returned by `PEAXAI_fitting()`. Depending on the class- balancing procedure, this dataset may contain only real observed DMUs or both real and synthetic DMUs. It defines the reference distribution for perturbing features.

importance_method — A named list (or data.frame-like) with the backend and its args:

- `name` One of `"SA"`, `"SHAP"`, `"LIME"`.
- `method` (SA) One of `"1D-SA"`, `"sens"`, `"DSA"`, `"MSA"`, `"CSA"`, `"GSA"`.
- `measures` (SA) e.g. `"AAD"`, `"gradient"`, `"variance"`, `"range"`.
- `levels` (SA) Discretization levels used by `rminer::Importance`.
- `baseline` (SA) Baseline value for SA, if applicable. If omitted, the DMU itself is used as baseline.
- `bg_n` (SHAP) Background sample size for `kernelshap::kernelshap` (default: 200).
- `n_permutations` (LIME) Number of permutations per observation (default: 5000).
- `feature_select` (LIME) Feature selection method, e.g., `"auto"`, `"none"`, `"forward_selection"` (default: `"auto"`).
- `bin_continuous` (LIME) Logical indicating if continuous features should be binned (default: `TRUE`).
- `n_bins` (LIME) Number of bins for continuous features (default: 4).

seed — Integer. Seed for reproducibility.

Details

Local explanations are produced for the predicted probability $\hat{p}_i = P(\text{efficient} \mid x_i)$. For `glm` models, the positive class is assumed to be the **second level** (`"efficient"`) and probabilities are extracted with `type = "response"`. For other models (e.g., **caret**), `predict(type = "prob")[, "efficient"]` is used.

The meaning of "local importance" depends on the backend:

- `"SHAP"` returns per-observation SHAP attributions; importances are computed from attribution magnitudes (e.g., `abs(SHAP)`) and normalized to sum to 1 per DMU.
- `"LIME"` fits a local surrogate linear model around the DMU using permutations from the reference distribution; the absolute weights of the local model are used as importance scores, normalized per DMU.
- `"SA"` computes local sensitivity of $\hat{p}_i$ to each feature around the DMU (using the chosen SA method/measure) and normalizes sensitivities per DMU.

Value

A numeric matrix (or data.frame) with one row per target observation and one column per predictor. Each row is a normalized importance profile whose values sum to 1. Column names match the predictor columns in x.

See Also

kernelshap, Importance, lime

Examples

```
data("firms", package = "PEAXAI")

# Firms of the Valencian Community: four inputs and one output
data <- subset(
  firms,
  autonomous_community == "Comunidad Valenciana"
)

x <- 1:4
y <- 5

models <- PEAXAI_fitting(
  data            = data,
  x               = x,
  y               = y,
  RTS             = "vrs",
  imbalance_rate  = NULL,
  methods         = list("glm" = list(weights = "dinamic")),
  trControl       = list(method = "cv", number = 3),
  metric_priority = c("Balanced_Accuracy", "ROC_AUC"),
  seed            = 1,
  verbose         = FALSE
)

final_model <- models[["best_model_fit"]][["glm"]]

# SHAP attributions via kernelshap. 'bg_n' is the background sample size
# (default 200); a smaller value keeps the example fast.
importance_method <- list(
  name = "SHAP",
  bg_n = 50
)

imp_local <- PEAXAI_local_importance(
  final_model       = final_model,
  x                 = x,
  y                 = y,
  explain_data      = data,
  reference_data    = final_model$trainingData,
  importance_method = importance_method,
  seed              = 1
```

```
)

# One row per DMU; the absolute weights of each row add up to 1
head(imp_local)
```

PEAXAI_peer *Identify Benchmark Peers Based on Estimated Efficiency Probabilities*

Description

Identifies peer units (i.e., reference benchmarks) for each decision-making unit (DMU) based on predicted probabilities of technical efficiency. Given a fitted classification model that estimates the probability of being efficient, the function selects, for each DMU, its nearest efficient peer according to Euclidean or weighted distances. Multiple efficiency thresholds can be specified to assess different levels of benchmarking stringency.

Usage

```
PEAXAI_peer(
  data,
  x,
  y,
  final_model,
  efficiency_thresholds,
  targets,
  weighted = FALSE,
  relative_importance = NULL
)
```

Arguments

`data`	A `data.frame` or `matrix` containing input and output variables used in the efficiency model.
`x`	Integer vector indicating the column indices of input variables in `data`.
`y`	Integer vector indicating the column indices of output variables in `data`.
`final_model`	A fitted classification model used to estimate efficiency probabilities. Supported classes: `"train"` (from **caret**) or `"glm"` (binomial).
`efficiency_thresholds`	Numeric vector indicating the minimum probability values required to consider a DMU as efficient.
`targets`	The object returned by `PEAXAI_counterfactuals`: a named list with one element per efficiency threshold, each containing the counterfactual projections (`counterfactual_dataset`) and the estimated β values (`inefficiencies`).

`weighted` Logical. If `TRUE`, peers are selected using weighted Euclidean distances based on variable importance. If `FALSE` (default), unweighted distances are used.

`relative_importance` Optional named numeric vector indicating the relative importance of each input/output variable (used when `weighted = TRUE`).

Details

This function enables probabilistic peer identification under uncertainty, supporting flexible definitions of efficiency based on thresholds over estimated probabilities. When `weighted = TRUE`, variable weights (e.g., derived from feature importance) modulate the peer selection process, allowing for context-aware benchmarking.

Value

A `data.frame` with one row per DMU. The first column, `DMU`, is the row index in `data`; the remaining columns are named after the efficiency thresholds and contain, for each DMU, the index of its closest efficient peer at that threshold. Distances are unweighted when `weighted = FALSE` and importance-weighted when `weighted = TRUE`.

Examples

```
data("firms", package = "PEAXAI")

# Firms of the Valencian Community: four inputs and one output
data <- subset(
  firms,
  autonomous_community == "Comunidad Valenciana"
)

x <- 1:4
y <- 5

models <- PEAXAI_fitting(
  data            = data,
  x               = x,
  y               = y,
  RTS             = "vrs",
  imbalance_rate  = NULL,
  methods         = list("glm" = list(weights = "dinamic")),
  trControl       = list(method = "cv", number = 3),
  metric_priority = c("Balanced_Accuracy", "ROC_AUC"),
  seed            = 1,
  verbose         = FALSE
)

final_model <- models[["best_model_fit"]][["glm"]]

# Peers are searched among the counterfactual projections, so the
# counterfactuals must be computed first.
relative_importance <- PEAXAI_global_importance(
```

```
  final_model       = final_model,
  x                 = x,
  y                 = y,
  explain_data      = data,
  reference_data    = final_model$trainingData,
  importance_method = list(name = "PI", n.repetitions = 5),
  seed              = 1
)

efficiency_thresholds <- c(0.75, 0.90)

targets <- PEAXAI_counterfactuals(
  data                  = data,
  x                     = x,
  y                     = y,
  final_model           = final_model,
  efficiency_thresholds = efficiency_thresholds,
  directional_vector    = list(
    relative_importance = relative_importance,
    baseline            = "mean"
  ),
  n_expand              = 0.5,
  n_grid                = 50,
  max_y                 = 2,
  min_x                 = 1
)

# Unweighted (Euclidean) peers
peers <- PEAXAI_peer(
  data                  = data,
  x                     = x,
  y                     = y,
  final_model           = final_model,
  efficiency_thresholds = efficiency_thresholds,
  targets               = targets,
  weighted              = FALSE
)

head(peers)

# Peers weighted by the global importance of each variable
peers_weighted <- PEAXAI_peer(
  data                  = data,
  x                     = x,
  y                     = y,
  final_model           = final_model,
  efficiency_thresholds = efficiency_thresholds,
  targets               = targets,
  weighted              = TRUE,
  relative_importance   = relative_importance
)
```

PEAXAI_predict *Predict Probability of Efficiency Using a Fitted Model*

Description

Predicts probabilities for new decision-making units (DMUs) using a fitted **caret** classification model. If `calibration_model` is provided, the raw classifier probabilities are post-processed to obtain calibrated probability estimates (e.g., Platt scaling via logistic regression or isotonic calibration via a monotone mapping).

Usage

```
PEAXAI_predict(data, final_model)
```

Arguments

`data`	A `data.frame` or `matrix` with the DMUs to be scored. It may carry any number of additional columns: only the variables `final_model` was trained on are kept, in the order the model expects. Those variables are matched by their **exact** name, so a variable **caret** renamed when the model was trained (`"Total assets"` becomes `"Total.assets"`) must already carry the sanitised name in `data`; otherwise it is reported as missing.
`final_model`	A fitted **caret** model provided by PEAXAI_fitting().

Value

A numeric vector of predicted probabilities (raw or calibrated), one per row of `data`.

Examples

```
data("firms", package = "PEAXAI")

# Firms of the Valencian Community: four inputs and one output
data <- subset(
  firms,
  autonomous_community == "Comunidad Valenciana"
)

x <- 1:4
y <- 5

models <- PEAXAI_fitting(
  data            = data,
  x               = x,
  y               = y,
  RTS             = "vrs",
  imbalance_rate  = NULL,
  methods         = list("glm" = list(weights = "dinamic")),
  trControl       = list(method = "cv", number = 3),
```

```
  metric_priority = c("Balanced_Accuracy", "ROC_AUC"),
  seed            = 1,
  verbose         = FALSE
)

final_model <- models[["best_model_fit"]][["glm"]]

# 'data' must contain every variable the model was trained on
probabilities <- PEAXAI_predict(
  data        = data[, c(x, y)],
  final_model = final_model
)

head(probabilities)
```

`PEAXAI_ranking` *Generate Efficiency Rankings Based on Probabilistic Classification*

Description

Produces efficiency rankings of decision-making units (DMUs) according to the probabilities estimated by a fitted classification model. Two ranking modes are supported:

- `"predicted"`: ranks DMUs solely by their predicted probability of being efficient.
- `"attainable"`: ranks DMUs hierarchically according to: (1) the attainable (target) efficiency probability, (2) the size of the improvement parameter β (smaller is better), and (3) the predicted efficiency probability (higher is better).

This allows to integrate both predictive and counterfactual (attainable) information into the efficiency ranking.

Usage

```
PEAXAI_ranking(
  data,
  x,
  y,
  final_model,
  efficiency_thresholds,
  targets = NULL,
  rank_basis
)
```

Arguments

`data`	A `data.frame` or `matrix` containing the input and output variables.
`x`	Integer vector specifying the column indices of input variables in `data`.
`y`	Integer vector specifying the column indices of output variables in `data`.
`final_model`	A fitted classification model used to estimate efficiency probabilities. Supported types are: • `"train"`: an object fitted with **caret**. • `"glm"`: a binomial logistic regression model.
`efficiency_thresholds`	Numeric vector defining one or more efficiency probability thresholds to determine the attainable frontier or peer set.
`targets`	The object returned by `PEAXAI_counterfactuals`: a named list with one element per efficiency threshold, each containing the counterfactual projections (`counterfactual_dataset`) and the estimated β values (`inefficiencies`). Only required when `rank_basis = "attainable"`.
`rank_basis`	Character string specifying the ranking criterion. Options are: • `"predicted"`: order units by predicted efficiency probability. • `"attainable"`: order by attainable probability, then by β, and finally by predicted probability (see Details).

Details

The attainable-based ranking combines predictive efficiency with the modeled potential for improvement (β) and the probability of reaching a target frontier level. This approach yields a more nuanced and interpretable prioritization of DMUs, reflecting both their current and achievable performance under the estimated model.

When `rank_basis = "attainable"`, ties in attainable probability are broken first by the magnitude of β (ascending), and then by the predicted probability (descending).

Value

- If `rank_basis = "predicted"`: a `data.frame` sorted by predicted efficiency probability.
- If `rank_basis = "attainable"`: a named list of `data.frames`, one per efficiency threshold, each sorted according to the hierarchical ranking scheme described above.

Examples

```
data("firms", package = "PEAXAI")

# Firms of the Valencian Community: four inputs and one output
data <- subset(
  firms,
  autonomous_community == "Comunidad Valenciana"
)

x <- 1:4
```

```
y <- 5

models <- PEAXAI_fitting(
  data            = data,
  x               = x,
  y               = y,
  RTS             = "vrs",
  imbalance_rate  = NULL,
  methods         = list("glm" = list(weights = "dinamic")),
  trControl       = list(method = "cv", number = 3),
  metric_priority = c("Balanced_Accuracy", "ROC_AUC"),
  seed            = 1,
  verbose         = FALSE
)

final_model <- models[["best_model_fit"]][["glm"]]

# (1) Ranking by predicted probability: neither 'efficiency_thresholds'
#     nor 'targets' are needed.
ranking <- PEAXAI_ranking(
  data        = data,
  x           = x,
  y           = y,
  final_model = final_model,
  rank_basis  = "predicted"
)

head(ranking)

# (2) Ranking by attainable probability: both arguments become mandatory.
relative_importance <- PEAXAI_global_importance(
  final_model       = final_model,
  x                 = x,
  y                 = y,
  explain_data      = data,
  reference_data    = final_model$trainingData,
  importance_method = list(name = "PI", n.repetitions = 5),
  seed              = 1
)

efficiency_thresholds <- c(0.75, 0.90)

targets <- PEAXAI_counterfactuals(
  data                  = data,
  x                     = x,
  y                     = y,
  final_model           = final_model,
  efficiency_thresholds = efficiency_thresholds,
  directional_vector    = list(
    relative_importance = relative_importance,
    baseline            = "mean"
  ),
  n_expand              = 0.5,
```

```
  n_grid                = 50,
  max_y                 = 2,
  min_x                 = 1
)

ranking_attainable <- PEAXAI_ranking(
  data                  = data,
  x                     = x,
  y                     = y,
  final_model           = final_model,
  efficiency_thresholds = efficiency_thresholds,
  targets               = targets,
  rank_basis            = "attainable"
)

head(ranking_attainable[["0.75"]])
```

`preprocessing` *Prepare Data and Handle Errors*

Description

This function arranges the data in the required format and displays some error messages.

Usage

```
preprocessing(data, x, y)
```

Arguments

`data`	A `data.frame` or `matrix` containing the variables in the model.
`x`	Column indexes of input variables in `data`.
`y`	Column indexes of output variables in `data`.

Value

It returns a `matrix` in the required format and displays some error messages.

reffcy *Random Sample for Efficiency Analysis*

Description

This function is used to simulate a `data.frame` with some given Data Generation Process.

Usage

```
reffcy(DGP, parms)
```

Arguments

`DGP`	Data Generation Process: • `"cobb_douglas_XnY1"`: Cobb-Douglas Data Generation Process for a XnY1 scenario. Check `help("cobb_douglas_XnY1")`.
`parms`	`list` with the parameters of the simulation functions. • `"N"`: Sample size. • `"nX"`: Number of inputs. Available for the `cobb_douglas_XnY1(N, nX, RTS)` function. • `"RTS"`: Returns to scale, `"vrs"` (default) or `"crs"`. Available for the `cobb_douglas_XnY1(N, nX, RTS)` function. • `"min_x"`: Lower bound of the (uniformly distributed) inputs (default: `1`). • `"max_x"`: Upper bound of the (uniformly distributed) inputs (default: `10`). • `"sd_u"`: Standard deviation of the inefficiency term (default: `0.4`). • `"sd_v"`: Standard deviation of the noise term (default: `0`, i.e. a deterministic frontier). Set to a value greater than `0` to obtain a stochastic frontier.

Details

Please refer to the help manuals of the mentioned functions for usage examples and more detailed information on the parameters.

Value

`data.frame` simulated with the selected Data Generation Process.

Examples

```
set.seed(1)

# 100 DMUs with 3 inputs and 1 output under variable returns to scale
simulated_data <- reffcy(
  DGP   = "cobb_douglas_XnY1",
  parms = list(
    N   = 100,
```

```
    nX  = 3,
    RTS = "vrs"
  )
)

# x1 ... x3 are the inputs, 'y' the observed output and 'yD' the
# theoretical frontier
head(simulated_data)

# A stochastic frontier is obtained by setting sd_v > 0
noisy_data <- reffcy(
  DGP   = "cobb_douglas_XnY1",
  parms = list(N = 100, nX = 1, RTS = "crs", sd_u = 0.4, sd_v = 0.1)
)

str(noisy_data)
```

`train_PEAXAI` *Training a Classification Machine Learning Model*

Description

This function trains a set of models and selects best hyperparameters for each of them.

Usage

```
train_PEAXAI(data, method, parameters, trControl, metric_priority, seed)
```

Arguments

`data`	A `data.frame` or `matrix` containing the variables in the model.
`method`	Parameters for controlling the training process (from the `'caret'` package).
`parameters`	A `list` of selected machine learning models and their hyperparameters.
`trControl`	A `list` of selected machine learning learning.
`metric_priority`	A `string` specifying the summary metric for classification to select the optimal model. Default includes `"Balanced_Accuracy"` due to (normally) unbalanced data.
`seed`	Integer. Seed for reproducibility.

Value

It returns a `list` with the chosen model.

`xai_prepare_sets` *Prepare Training and Target Datasets from a caret Model*

Description

Extracts and formats the training and/or target datasets from a machine learning model trained with `caret::train`, allowing for distinction between using the full training data or only the original subset used for modeling. It standardizes the class column to be named `"class_efficiency"` and positions it as the last column.

Usage

```
xai_prepare_sets(
  data,
  x,
  y,
  final_model,
  background,
  target,
  type,
  threshold,
  levels_order
)
```

Arguments

`data`	A `data.frame` containing the original dataset used to train the model. Only needed when using `"real"` as background or target.
`x`	Not currently used. Reserved for future input variable selection.
`y`	Not currently used. Reserved for future output variable specification.
`final_model`	A trained model object of class `"train"` from the **caret** package.
`background`	A character string, either `"train"` or `"real"`, specifying the background dataset used for explainability.
`target`	A character string, either `"train"` or `"real"`, specifying the target dataset to be explained.
`type`	Not currently used. Reserved for future prediction types.
`threshold`	Not currently used. Reserved for future thresholding logic.
`levels_order`	A character vector specifying the levels of the response factor, typically `c("not_efficient", "efficient")`. Not currently used, but can help in reordering or relabeling.

Value

A `list` with two elements:

`train_data` A `data.frame` representing the background dataset, with the class column renamed to `"class_efficiency"` and positioned last.

target_data A data.frame representing the target dataset, formatted in the same way.

Index

∗ **datasets**
data, 3
data_example, 4
data_SABI, 5
firms, 7

convex_facets, 2

data, 3
data_example, 4
data_SABI, 5

FeatureImp, *17*
find_beta_maxmin, *11*
firms, 7

get_SMOTE_DMUs, 8

Importance, *17*, *20*

kernelshap, *17*, *20*

lime, *20*

npudensbw, *10*

PEAXAI_counterfactuals, 10, *21*, *26*
PEAXAI_fitting, 13
PEAXAI_global_importance, 15
PEAXAI_local_importance, 18
PEAXAI_peer, 21
PEAXAI_predict, 24
PEAXAI_ranking, 25
preprocessing, 28

reffcy, 29

train, *11*
train_PEAXAI, 30

xai_prepare_sets, 31